
# NAVAL POSTGRADUATE SCHOOL

**MONTEREY, CALIFORNIA**

# THESIS

**THE IMPACT OF SOFTWARE REUSE ON THE COST OF NAVY SONAR AND FIRE CONTROL SYSTEMS**

by

Anthony M. Wilson

June 2009

| | |
|---|---|
| Thesis Advisor: | Joseph G. San Miguel |
| Second Reader: | Michael W. Boudreau |

**Approved for public release; distribution is unlimited**

THIS PAGE INTENTIONALLY LEFT BLANK

| REPORT DOCUMENTATION PAGE | | | *Form Approved OMB No. 0704-0188* |
|---|---|---|---|
| Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503. | | | |
| **1. AGENCY USE ONLY** ***(Leave blank)*** | **2. REPORT DATE** June 2009 | **3. REPORT TYPE AND DATES COVERED** Master's Thesis | |
| **4. TITLE AND SUBTITLE** The Impact of Software Reuse on the Cost of Navy Sonar and Fire Control Systems | | | **5. FUNDING NUMBERS** |
| **6. AUTHOR(S)** Anthony M. Wilson | | | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Naval Postgraduate School Monterey, CA 93943-5000 | | | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| **9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)** N/A | | | **10. SPONSORING/MONITORING AGENCY REPORT NUMBER** |
| **11. SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. | | | |
| **12a. DISTRIBUTION / AVAILABILITY STATEMENT** Approved for public release; distribution is unlimited | | | **12b. DISTRIBUTION CODE** A |

**13. ABSTRACT (maximum 200 words)**

One of the critical aspects in the design and sustainment of new and replacement Navy combat systems is the development of software to run the systems in a manner that maximizes their benefit to national security. This research examines the Navy's acquisition of anti-submarine warfare sonar and fire control software to determine if software reuse has been effective in lowering costs. The potential for cost avoidance exists due to the commonality of the anti-submarine warfare mission across the surface, air, surveillance, and submarine communities. The three categories of costs chosen for analysis are maintenance; training; and research, development, test, and evaluation (RDT&E). Analysis focuses on the identification of trends associated with each of the costs for selected systems and programs. Identifying trends in funding could provide evidence of the cost-effectiveness of software reuse efforts within and across the surface, air, surveillance, and submarine communities.

| **14. SUBJECT TERMS** Software reuse, reuse, ASW software, sonar, fire control, APB, ARCI | | | **15. NUMBER OF PAGES** 121 |
|---|---|---|---|
| | | | **16. PRICE CODE** |
| **17. SECURITY CLASSIFICATION OF REPORT** Unclassified | **18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified | **19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified | **20. LIMITATION OF ABSTRACT** UU |

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

**Approved for public release; distribution is unlimited**

# THE IMPACT OF SOFTWARE REUSE ON THE COST OF NAVY SONAR AND FIRE CONTROL SYSTEMS

Anthony M. Wilson
Lieutenant, United States Navy
B.S., United States Naval Academy, 2003

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF BUSINESS ADMINISTRATION**

from the

**NAVAL POSTGRADUATE SCHOOL**
**June 2009**

Author: Anthony M. Wilson

Approved by: Joseph G. San Miguel
Thesis Advisor

Michael W. Boudreau
Second Reader

William Gates
Dean, Graduate School of Business and Public Policy

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

One of the critical aspects in the design and sustainment of new and replacement Navy combat systems is the development of software to run the systems in a manner that maximizes their benefit to national security. This research examines the Navy's acquisition of anti-submarine warfare sonar and fire control software to determine if software reuse has been effective in lowering costs. The potential for cost avoidance exists due to the commonality of the anti-submarine warfare mission across the surface, air, surveillance, and submarine communities. The three categories of costs chosen for analysis are maintenance; training; and research, development, test, and evaluation (RDT&E). Analysis focuses on the identification of trends associated with each of the costs for selected systems and programs. Identifying trends in funding could provide evidence of the cost-effectiveness of software reuse efforts within and across the surface, air, surveillance, and submarine communities.

THIS PAGE INTENTIONALLY LEFT BLANK

# TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF FIGURES

# LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| ACAT | Acquisition Category |
| ACB | Advanced Capability Build |
| APB | Advanced Processing Build |
| ARCI | Acoustic Rapid COTS Insertion |
| ASN | Assistant Secretary of the Navy |
| ASW | Anti-submarine warfare |
| BA | Budget Authority |
| BLS | Bureau of Labor Statistics |
| CAL | Common Asset Library |
| CBO | Congressional Budget Office |
| COCOMO | Constructive Cost Model |
| COI | Community of Interest |
| COTS | Commercial off the shelf |
| FM&C | Financial Management & Comptroller |
| GAO | Government Accountability Office |
| NCCA | Naval Center for Cost Analysis |
| NETPDTC | Naval Education and Training Professional Development Technology Center |
| NOA | Naval Open Architecture |
| OAML | Oceanographic and Atmospheric Master Library |
| RDT&E | Research, development, test, and evaluation |
| SEI | Software Engineering Institute |
| SLOC | Source Lines of Code |
| SRDR | Software Resource Data Report |
| STRG | Submarine Tactical Requirements Group |
| TEASG | Test, Evaluation, and Assessment Support Group |
| TI | Technology Insertion |
| VAMOSC | Visibility and Management of Operating and Support Costs |

THIS PAGE INTENTIONALLY LEFT BLANK

# ACKNOWLEDGMENTS

I am thankful to all those past and present that have provided for the defense of our great country. Opportunities such as this would not be available without your sacrifice. To my wife and best friend, Katie: you tolerated many sleepless nights and supported me all the way. I love you and cannot thank you enough for all that you do. To my advisors, Dr. Joseph San Miguel and Professor Michael Boudreau: your wisdom and ability to keep things in perspective kept me on the right track. I would not have been successful without your help. Finally to CAPT Charles Davis, Jim Thompson, Colleen Cannon, Bill Johnson, and Steve Oxman: your willingness to help and contributions are sincerely appreciated.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

## A. PURPOSE OF THE STUDY

The purpose of this study is to evaluate whether the reuse of sonar and fire control related software in anti-submarine warfare systems has been beneficial in reducing associated costs of maintenance, training, and research and development. The reuse of software is examined as an alternative to developing all aspects of a software system from scratch for one time use. Anti-submarine warfare (ASW) software was chosen due to the commonality of the ASW mission in the submarine, surface, aviation, and surveillance communities. If there is a correlation between the methods in which ASW software is developed and reused, and a reduction in the cost of developing and maintaining ASW software systems, then there may be strong support for building systems that increase the capacity for software reuse. The objective of this research is to determine if actual cost savings have resulted from ASW software reuse.

## B. THE CHALLENGE

The Department of Defense routinely acquires complex and advanced weapons systems such as the Virginia Class submarine, the Littoral Combat Ship, and the Future Combat System. To operate and run these systems, a significant investment in software programs is usually required. The necessity to acquire these systems quickly and within specified costs makes software development even more difficult and critical to successful implementation. A 2004 Government Accountability Office [GAO] report estimated that during fiscal year 2003, the Department of Defense spent close to forty percent of its research, development, test, and evaluation (RDT&E) budget on software. This equates to approximately $21 billion. Given the Congressional Budget Office's estimate of a $1.7 trillion dollar deficit for 2009 (Congressional Budget Office [CBO], 2009) and increasing pressure on the Navy budget, any cost savings on software programs should be considered beneficial. GAO (2004) also states that a survey of software systems developed in the 1990s illustrated that on average software systems had a 189 percent

cost overrun, a 222 percent schedule overrun, and delivered only 61 percent of the anticipated capabilities. Certainly, improvements in these factors would also save costs and help achieve mission requirements.

To counteract these trends, the Navy now emphasizes the development of software systems using an Open Architecture and modular design. Open architecture allows for greater "interchangeability" of software components, a lower barrier of entry for future development by software companies, and more rapid increases in technology (Nelson, 2007, p. 8). The concept of Naval Open Architecture (NOA) is defined as follows in Program Executive Office—Integrated Warfare Systems 7 [PEO IWS 7] (2007).

> Naval Open Architecture (NOA) is the confluence of business and technical practices yielding modular, interoperable systems that adhere to open standards with published interfaces. This approach significantly increases opportunities for innovation and competition, enables re-use of components, facilitates rapid technology insertion, and reduces maintenance constraints. NOA delivers increased warfighting capabilities in a shorter time at reduced cost. (p. 2)

Anti-submarine sonar and fire control software developed within the Acoustic Rapid COTS (commercial-off-the-shelf) Insertion (ARCI) program and Advanced Processing Build (APB) process is an example of such an acquisition and development strategy. The ARCI/APB program began in 1996, and has since succeeded in achieving commonality of sonar software systems within the submarine community. Its efforts have also led to the opportunity for sharing and reuse of software components among the air, surface, and surveillance communities.

Three anti-submarine warfare systems were chosen to investigate the potential effects of software reuse on maintenance and training costs. The reasons for selecting these systems are discussed in Chapter III. The three systems are:

- AN/BQQ-10 Sonar System
- AN/BYG-1 Submarine Combat Control System
- AN/SQQ-89 Surface Combat System

Additionally, there are several RDT&E budget items used to investigate whether reuse of ASW software has resulted in a decrease in funding for the development and testing of ASW related software systems. Budget items were selected for reasons discussed in Chapter III. The following RDT&E budget items are listed according to their budget activity, program element number, program element name, project number and project name.

- Budget Activity (BA)—4, Program Element 0603561N/Advanced Submarine System Development—Project Unit 0223/Sub Combat System Improvement (ADV)
- Budget Activity (BA)—4, Program Element 0603553N/Surface ASW—Project Unit 1704/Undersea Warfare
- Budget Activity (BA)—5, Program Element 0604503N/Submarine System Equipment Development—Project Unit 0219/Sub Sonar Improvement (ENG)
- Budget Activity (BA)—5, Program Element 0604562N/Submarine Tactical Warfare System—Project Unit 0236/SSN Comb Cont Sys Imprvmnt (ENG)
- Budget Activity (BA)—5, Program Element 0604261N/Acoustic Search Sensors—Project Unit 0480/ASW Sensors & Processors
- Budget Activity (BA)—7, Program Element 0205620N/Surface ASW Combat System Integration—Project Unit 0896/AN/SQQ-89 Modification and Budget Activity (BA)—7, Program Element 0205620N/Surface ASW Combat System Integration—Project Unit 1916/Surface ASW System Improvement

Cooperation, assistance, and data were provided by cost estimators, engineers, acquisition professionals, and program managers from the Program Executive Office (PEO) for Integrated Warfare Systems, Program Executive Office (PEO) for Submarines, Naval Center for Cost Analysis (NCCA), Air Force Center for Cost Analysis (AFCCA), Naval Air Systems Command (NAVAIR), Office of Naval Research (ONR), General Dynamics (GD), Lockheed Martin Maritime Systems and Sensors Division, BAE Systems, Joint Networking Technologies, and Advanced Acoustic Concepts (AAC).

In addition to the commonality of the ASW mission across the air, surface, surveillance, and submarine communities, decoupling hardware and software in the

design of anti-submarine warfare sonar and fire control systems has permitted a greater opportunity for reuse of software across these communities. With each program developing updates on a regular basis, the opportunity for reuse is also accentuated.

Reuse benefits that have been documented include lower development costs and improved quality, which should ultimately lower maintenance and support costs. Thus there should be trends that indicate reduced RDT&E and maintenance and support costs. Additionally, if reuse is prominent across each community, there may be a correlation between funding profiles. By determining the trends in these associated costs, this research should assess whether software reuse, which has been touted as means of cost avoidance, has benefitted the Navy.

## C. RESEARCH OBJECTIVES

By determining cost savings and trends in funding, this research could make the following contributions to future and current defense software acquisitions.

- Highlight successful programs that can serve as models for other defense acquisition programs. The techniques and efforts employed to reduce costs could potentially be implemented in similar programs to lower acquisition and support costs.
- Illustrate whether current practices and methods being employed to foster the reuse of software are effective and should be continued. If current practices are not successful, then there should be an evaluation of whether they should continue in the same manner.

In attempting to make these contributions the following research questions were posed.

- What type of efforts has the Navy undertaken to reuse ASW software?
- What cost data should be analyzed to best represent the potential savings that could be realized through the reuse of ASW software?
- Has software reuse been successful in reducing the costs of developing and maintaining ASW sonar and fire control systems?

## D. THESIS ORGANIZATION

This thesis is organized into five chapters. Chapter I introduces the purpose of this study. It also highlights the motivation behind the research for both the development of sonar and fire control software and the subsequent reuse. If positive trends can be identified, it may provide other communities with insight into how cost reduction can occur in developing software.

### 1. Chapter II—Background

The information used in this chapter was gathered through an extensive literature review of software reuse practices and interviews with personnel involved in the design, costing, and procurement of software intensive systems. Personnel that contributed came from a variety of organizations including;

- Program Executive Office (PEO) for Submarines, Washington, DC
- Program Executive Office (PEO) for Integrated Warfare Systems, Washington, DC
- Naval Center for Cost Analysis (NCCA), Washington, DC
- Naval Air Systems Command (NAVAIR), Patuxent River, MD
- Air Force Center for Cost Analysis, Waltham, MA
- Joint Networking Technologies, LLC
- Advanced Acoustic Concepts, Columbia, MD
- BAE Systems, Washington, DC
- General Dynamics, Fairfax, VA
- Lockheed Martin Maritime Systems and Sensors, Manassas, VA

Chapter II begins by discussing the submarine community's process for developing sonar and fire control software. It continues with a discussion of how current system design involving the separation of operations into functional components as well as a goal towards commonality has led to greater opportunities for reuse of software assets within the submarine, surface, air, and surveillance communities.

The next section defines software reuse and provides background concerning what types of assets used in the design of software intensive systems can be reused and which are primarily referred to when describing reuse.

The final section provides additional detail concerning the effect of software reuse on costs as well as some of the strategic factors involved in successful reuse programs. This includes examples of reuse practices that have been successful as well as the constraints faced when organizations attempt to reuse software. Also included in this section is a discussion of the potential benefits that organizations can expect when reusing software assets.

### 2. Chapter III—Methodology

The Methodology Chapter describes how data were retrieved, normalized, and analyzed. Data related to maintenance and training costs were provided by the Navy Visibility and Management of Operating and Support Costs (VAMOSC) database. This database is maintained by the Naval Center for Cost Analysis (NCCA).

RDT&E budget data were obtained from the Assistant Secretary of the Navy (ASN) Financial Management & Comptroller (FM&C) website, http://www.finance.hq.navy.mil/FMC/. The data were normalized using the RDT&E (purchase) appropriation from the Naval Center for Cost Analysis (NCCA) inflation calculator (February 2009 edition).

Data related to the wages for software engineers and computer programmers was obtained from the Occupational Employment Statistics (OES) section of Bureau of Labor Statistics (BLS) website http://www.bls.gov/. The data were normalized using the Consumer Price Index (CPI) calculator.

### 3. Chapter IV—Results

The Results Chapter contains descriptions of the outputs from analysis conducted in the Methodology Chapter. Trend analysis of the cost data is displayed graphically and within tables, and is accompanied by a narrative to describe the results and any unusual circumstances. Possible reasons for the various trends identified are also included in the analysis and narrative.

### 4. Chapter V—Summary, Conclusions, and Recommendations

The Summary of Results section of Chapter V provides an overview of the results obtained in Chapter IV. It gives a general description of the trends exhibited by the various costs that were analyzed.

In the Conclusions section, judgments were made concerning the relationship of the data analyzed to software reuse cost avoidance.

The Recommendations section provides potential areas for future research and suggestions to improve future analysis.

THIS PAGE INTENTIONALLY LEFT BLANK

# II. BACKGROUND

## A. ACOUSTIC RAPID COTS INSERTION

In the mid 1990s, the United States submarine force faced a difficult situation. In the face of significant budget cuts resulting from the end of the Cold War, the acoustic advantage that the United States held over its Russian counterparts was eroding (Boudreau, 2006). Traditionally, this would have resulted in a "multi-billion dollar development program stretched over 12 or more years" to design a better system (Johnson, 2004, p. 100). However, due to the economic environment, a new strategy was needed in order to provide the maximum capability at the lowest possible cost. The result was an evolutionary acquisition strategy known as the Acoustic Rapid COTS (commercial-off-the-shelf) Insertion (ARCI) program and the Advanced Processing Build (APB). The plan for ARCI was approved by the Milestone Decision Authority in 1996, and the first ARCI upgrade was provided to the fleet in November of 1997, 18 months later. This was a significantly shorter time than the six-and-a-half years required for the previous system, the AN/BSY-2 combat system.

ARCI seeks to improve sonar systems in the submarine force through the use of commercial technology and planned upgrades to take advantage of advances in technology. The two types of upgrades utilized within the submarine force are known as technology insertions (TIs) and advanced processing builds (APBs), and are described in the sections below.

The ARCI goal is to leverage advances in commercial technology to provide leading-edge products to fleet end users. This is achieved not only through design but through the opening of competition from a variety of sources, including small business, universities, government labs, and traditional defense contractors.

### 1. Technology Insertions

Recognizing the rapidly changing pace of technology, ARCI seeks to take advantage of advances in processor capability. To accomplish this goal, ARCI

implemented a process known as Technology Insertions (TI). Technology insertions are hardware-only updates to a system. They are provided every two years and establish a new hardware baseline for future upgrades. This allows ASW systems to take advantage of the greater processing capacity afforded by commercial advances. They are numbered according to the year of development. An example is TI–04. This means that the hardware baseline for that ship's system was completed in 2004.

### 2. Advanced Processing Builds

One of the innovative approaches to the ARCI program involves the use of a design architecture that allows developers to "decompose new systems along natural and logical boundaries, at the functional string and thread level, to enable focused, iterative design and assessment" (Johnson, 2004, p. 100). The application software within ARCI is isolated into functional modules which can then either "stand-alone or can be re-used and installed in another system application" (Johnson, 2004, p. 100). Ultimately, this allows modules of software developed for submarines to then be used on other hardware systems, such as those employed by the surface fleet. In addition, the software used within an ARCI system is hardware independent, and improvements can be made to software applications independent of changes in hardware. Within the ARCI program, these software improvements are referred to as Advanced Processing Builds (APBs).

APBs are managed by the Program Executive Office for Integrated Warfare Systems 5A (PEO IWS 5A). APBs are "hardware independent software builds to create or improve functionality for transition to naval combat systems programs" (Program Executive Office-Integrated Warfare Systems 5A [PEO IWS 5A], 2003, p. 8). PEO IWS 5 (2009) refers to APBs as a "process as well as a product" (p. 1). APBs are developed on an 18 to 24-month cycle and provide annual deliveries to the fleet (PEO IWS 5, 2003, p. 3). They are numbered based upon the year of development. An example is APB–04. This means that the software baseline for that system was completed in 2004. The ultimate baseline for a system can be referenced by the TI and APB

baselines. Based on the two examples provided, a ship that has a baseline of TI-04 and APB-04 contains the hardware and software baselines developed in 2004. The process used to develop APBs is described below.

The process begins through an "understanding of fleet needs" (PEO IWS 5, 2009, p. 1) as provided by OPNAV N87 and the APB Development letter (PEO IWS 5, 2009, p. 2). The capability needs of the fleet contained within the APB Development letter are generated by the Submarine Tactical Requirements Group (STRG). The STRG consists of Navy captains in "key requirements positions" (PEO IWS 5, 2009, p. 2) that identify tactical and capability needs within the submarine force (PEO IWS 5, 2009, p. 2). After review by submarine force leadership, OPNAV N87 provides PEO IWS 5A with the APB Development letter that contains both the requirements and metrics that must be met in order to institute the new capability (PEO IWS 5, 2009, p. 2). Further reviews are conducted once potential capabilities are matched with available technology. When the final development letter is provided, PEO IWS 5A provides the technology needs to the following types of organizations:

- University Labs such as the Johns Hopkins Applied Physics Laboratory (JHU-APL) and the Applied Research Laboratory at the University of Texas (ARL/UT)
- Government laboratories such as the Office of Naval Research (ONR), the Defense Advanced Research Projects Agency (DARPA), and the Naval Undersea Warfare Center (NUWC)
- Small businesses
- Defense contractors such as General Dynamics and Lockheed Martin

There are four steps involved in the development of software for APBs. Step one of the APB process is technology evaluation and involves the evaluation of products developed by the organizations listed through a peer review process. According to PEO IWS 5 (2009),

> Peer review is the process by which experts from the Navy and university labs, developers, scientists, and engineers examine technologies in an objective manner, each member bringing his or her expertise to bear. It is

> important that members are from many organizations, that developers are included, that competition is fair and unbiased, and that entry into the process is open to all qualified applicants. (p. 8)

The experts selected to examine the products evaluate the current technology and recommend algorithms for step two, for revision and further review, for deferral to a future APB, or for rejection (PEO IWS 5, 2009, p. 5).

Step two tests mature technologies "that show promise of providing performance improvements to satisfy requirements" (PEO IWS 5, 2009, p. 5). Algorithms from step one are tested in a computing laboratory environment. The results are then evaluated and technologies are recommended for further development and testing (PEO IWS 5, 2009, p. 5).

Step three integrates the technologies into a system to test the functionality on the most current hardware available. Testing in Step three "measures performance of individual algorithms as well as the integrated system using real and simulated data to ensure that the product addresses requirements and provides a measurable return on investment" (PEO IWS 5, 2009, p. 6). A report on the results is prepared by the Test, Evaluation, and Assessment Support Group (TEASG) to recommend the new algorithms for step four.

Step four is performed in odd years of the APB, and consists of at-sea-testing. Results from step four are provided to PEO IWS 5A, which then make a recommendation of the algorithms tested for production (PEO IWS 5, 2009, p. 6). Algorithms accepted for production are then provided to applicable program offices within the Program Executive Office (PEO) for Submarines.

Final production is accomplished by the system integrator. For acoustic-related software, Lockheed Martin's Maritime Systems and Sensors Division, located in Manassas, VA, is responsible for final integration and production of a new APB baseline. However, production does not begin only after the completion of step four. Production is

an ongoing process, begins early in the development cycle (PEO IWS 5, 2009, p. 4), and involves the system integrator and the organizations that develop the final technology products.

The APB process recognizes that the "development of adaptive, complex systems requires an iterative design and development approach" (Johnson, 2004, p. 100). Additionally PEO IWS 5 (2009, p. 12) cites several principles that affect its overall success.

- No one organization has the complete answer.
- Small businesses are key to innovation. Their input must be sought, and their viability must be sustained.
- An open process exists, based on mutual sharing of development materials, where legitimate intellectual properties are protected.
- Cooperation among participants in the development process is essential to delivering a quality product in the time frame available.
- Fair testing, conducted by independent evaluators, using real data and fleet operators, and where possible sea testing is needed to ensure that the APB is effective.
- Fleet input is sought and used throughout the development cycle.
- All participants are governed by the same set of rules.

#### *a. Advanced Capability Builds*

The surface community employs a similar process known as the Peer Review Process (Clements, Cohen, and Bergey, 2006, p. 21). It also utilizes a peer review process to develop and integrate new technology. The upgrades created by the process are known as Advanced Capability Builds (ACBs). In the future, the intent is to incorporate a software library into the process as well. The library is known as the Common Asset Library (CAL). The CAL will serve as a repository for components that will later be integrated into Advanced Capability Builds (P. Gill, personal communication, March 23, 2009).

### *b. Advanced Processing Builds and Software Reuse*

The APB process is one of the primary methods of software development for the submarine community. Although initially meant to provide only improvements to sonar systems, APBs now incorporate improvements to tactical systems such as fire control systems. Additionally, APBs are cumulative. That is, they build upon previous versions. Thus, when a system is updated it not only gets the current version of the APB, but all of the previous upgrades as well. In terms of software reuse, this is a key consideration, specifically when viewing an APB as a product. They are designed to be easily "ported" from one system to another with few code changes (Clements et al., 2006, p. 29).

### *c. Funding*

Although APBs provide systems with updated technology, they require a steady stream of research, development, test, and evaluation (RDT&E) funding to produce annual upgrades and establish new APB baselines. Once APBs and TIs are developed, they must also be purchased and installed using Other Procurement, Navy (OPN) funds.

APB updates are not provided to each submarine annually. The goal is to update a percentage of fleet systems on an annual basis. This percentage may vary depending upon funding and ship schedules. There is currently a wide range of APB and TI baselines within the fleet, from TI-98 to TI-08 and APB-00 to APB-07. So, despite having hardware baselines developed every two years and software every year, there are some units operating with hardware baselines that are ten years old and software that is eight years old. This is still a better alternative to previous methods of development and system upgrade that allowed some ships to operate with technology that may have been fifteen to twenty years old.

## 3. Commonality

Anti-submarine warfare is a complex national security mission that requires coordination among various communities within the Navy. An inability to find and

prosecute enemy submarines can cause significant losses to Navy assets and severely disrupt the ability to control the seas during a time of war (Benedict, 2005, p. 93-94). Unifying the efforts of each of the communities with respect to "vision, acquisition strategy, and the organization and resources needed to implement them" provides greater assurance that required capabilities are implemented (Benedict, 2005, p. 95). Within the Navy, the anti-submarine warfare mission is common to the submarine, surface, aviation, and surveillance communities.

In order to achieve commonality of fleet wide submarine sonar systems, ARCI implemented a phased approach to replace and upgrade existing systems. This goal has largely been achieved (Clements et al., 2006, p. 17). In addition to realizing the potential for commonality of sonar systems across the submarine community, the separation of sonar system functions presents an opportunity for greater commonality across other communities such as surface, air, and surveillance (Clements et al., 2006, p. 15). Sonar systems in their basic form consist of hydrophones, arrays, inboard processors and processing, and operator displays (Clements et al., 2006, p. 14).

Operation of sonar systems can be broken down further and described as the processing of acoustic signals detected by sensors and displayed in some form to an operator (Clements et al., 2006, p. 14). Acoustic signals are detected by arrays consisting of hydrophones and transducers used to detect radiated noise. Each type of array detects a certain type of acoustic signal (Clements et al., 2006, p. 15). In general, the types of acoustic signals are broken down into ranges of frequencies and are classified as either low frequency (LF), medium frequency (MF), or high frequency (HF) (Clements et al., 2006, p. 15). Just as the arrays themselves differ in structure, the processing required to "translate" the signals into usable operator output differs (Clements et al., 2006, p. 15). The potential for commonality and software reuse across sonar systems involves uniform processing between the three ranges of frequencies as opposed to the individual arrays themselves (Clements et al., 2006, p. 15).

The operation of fire control systems can be similarly broken down. A fire control system takes data from sonar or other inputs, tags the data, stores them, and then creates a target solution that can be used to fire weapons. Although there may be differences in the source of the data, the functionality of the system remains the same.

By breaking down these software intensive systems into their functional modules, the opportunity to reuse them in other applications is enhanced. Systems on other platforms that have the same capability requirements could potentially reuse components. Specific metrics evaluating the extent to which this is accomplished are not well documented. However, anecdotal evidence from program offices and contractors confirms that ASW software is reused within and between the submarine, surface, air, and surveillance communities. Software reuse is discussed in further detail in the following sections.

## 4. Summary

The goal of the ARCI program is to provide the submarine community with improved system performance at lower costs (Naval Sea Systems Command [NAVSEA] PMS 4252, 1999). A study conducted in 2006 comparing the costs of the ARCI strategy of development and acquisition to legacy sonar systems (ASSET, 2006) found the following results:

- 2.1 to 1 reduction in budget allocation across Shipbuilding and Conversion, Navy (SCN); Other Procurement, Navy (OPN); Operations and Maintenance, Navy (O&MN); Research, Development, Test, and Evaluation (RDT&E); and Military Construction (MilCon)
- 6 to 1 reduction in contract dollars for development and production
- 8 to 1 reduction in operating and support costs

The researcher was unable to validate these data. However, the results have been validated by NAVSEA. The depth of this report is outside the scope of this research and simply provides an indication of the success of the ARCI program. Finally, the cost reductions described are not software exclusive and take all aspects of a sonar system into account, including hardware.

## B. SOFTWARE REUSE CONSIDERATIONS

Software reuse is not a new concept within the Department of Defense. Government Accountability Office [GAO] (1993) analyzed the Department of Defense's plans to take advantage of software reuse in an effort to lower acquisition costs. At the time, the Department of Defense estimated that it was spending in excess of $24 billion per year on software related expenses (GAO, 1993, p. 1). The report cites a draft of the Department of Defense's technology strategy from 1991, where a savings of $11.3 billion is estimated over a fifteen-year period due to software reuse (GAO, 1993, p. 19). The accuracy of this estimate is not known, but the report illustrates the prolonged interest in software reuse.

The potential benefits of software reuse are well documented in the literature (Karlsson, 1995; Mili, Mili, Yacoub, and Addy, 2002; Poulin, Caruso, and Hancock, 1993). Potential benefits include reduced cost, greater productivity, shorter development time, and increased quality. Strategies and methods of determining the potential benefits of reuse are also the subject of numerous studies (Poulin et al., 1993; Rothenberger, Dooley, Kulkarni, and Nada, 2003). A finding that is common to several studies comes from Nelson (2007):

> If reuse is to be of substantial value to an enterprise, it will require management. The enterprise will have to establish some method by which reusable components are proposed, validated and made available; it will also need to establish methods to assure that new projects make the most use of existing reusable assets. This can present an organization with challenges in how to share the costs, ownership and cross-organizational responsibilities. (p. 27)

This quote provides a baseline of the factors involved in taking full advantage of software reuse in the large-scale development of software systems. It illustrates the necessity for combining both technical knowledge and managerial skill when attempting to apply software reuse to the development process.

### 1. Definition of Reuse

As mentioned above, software reuse can be complicated and may encompass a wide range of strategies. Thus the concept of software reuse requires a broad definition. Karlsson (1995) defines software reuse as the "process of creating software systems from existing software assets rather than building software systems from scratch" (p. 3). PEO IWS 7 (2007) defines software reuse as "the process of implementing or updating software systems using existing assets" (p. 10–9). Reuse does not simply refer to the recycling of the same system. A component can be used many times, but only reused one time per application (Poulin et al., 1993, p. 576). In other words, if a software asset is reused to create multiple copies of the same system, each copy is not considered an example of reusing software, only the original is.

It is important not to confuse reuse with recycling of software assets. To classify as reusable, a software component must be capable of being used in other applications that are similar. Determining instances that facilitate the reuse of a component are important in the reuse process. Further guidance is provided within the literature and texts concerning what products associated with software development should be considered candidates for reuse.

### 2. Reusable Assets

Software code is typically the most common asset involved in reuse (Mili et al., 2002, p. 7). However, it is worth noting that there are other assets associated with software development that are considered reusable. Classifying the types of assets that can be considered reusable is a key aspect of the practice of software reuse. Having a better understanding of all of the assets that can be considered for reuse allows organizations to better determine how to take advantage of reuse. The assets listed below

are not comprehensive, but represent a general description of the types of assets that are available when making the decision to reuse software. The list is taken from Mili et al., (2002), but other references such as Poulin et al., (1993) and Karlsson (1995) contain similar descriptions of assets. PEO IWS 7 (2007) provides descriptions of assets that are tailored to the specific aspects of Department of Defense software development and testing processes. Mili et al., (2002, pp. 7–9) covers these assets and how they can be classified on a more general basis.

- **Executable Code:** The essence of a piece of executable code is the function that it computes; executable code is typically represented in machine-readable form, and is indexed by means of its functional properties.
- **Source Code:** Source code embodies a function. But to the extent that it also embodies structural information, source code can be viewed as problem-solving knowledge. Source code is represented by programming languages, and can be indexed by means of its structural properties, as well as its functional properties.
- **Requirements Specification:** Requirements specifications are descriptions of the systems being developed. They are the products of eliciting user requirements and recording them in some notation. Specification can be represented in natural language, in formal notation, or in a mixture thereof. Specifications are indexed by means of the functional properties that they capture, and may be reused to build either compound specifications or variations on the original product.
- **Designs:** Designs are generic representations of design decisions. Their essence is the design/problem-solving knowledge that they capture. In contrast to code assets, designs are not executable. In contrast to specification assets, they capture structural information rather than functional information. They are represented by patterns that can be instantiated in different ways to produce concrete designs. Unlike functions or modules, designs cannot be indexed by their functional properties; rather they can be indexed by features of the family of problems that they solve.
- **Test Data:** Once a software system has been designed it typically must be integration-tested using some test data. These test data can be reused to test the system following some action such as maintenance or can be reused to test a similar product which has a similar set of inputs but different output conditions. Representation of these data is straightforward, and the data can be indexed by a description of the input domain of the software system or some general indication of the function of the system.

- **Documentation:** Natural-language documentation that accompanies an asset can also be a reusable asset. Documentation is most typically represented in natural language and can be indexed via the asset that it documents.
- **Architectures:** Software architecture defines the structure of a software system as the aggregate of a set of components that exchange data. The constructs by which building blocks are usually combined in an architecture have a higher level of abstraction than do programming language constructs, and are of a different nature. They prescribe information flow, control flow, or communication protocols between components. Architectures are represented by means of specialized notations, and are indexed by means of their architectural features.

## 3. Summary

Illustrating the number and variety of assets that can be considered for reuse shows that there is not a simple answer to the question of how to achieve the maximum benefit from the reuse of software assets. Reusing software assets is not the same as reusing other assets in an engineering process (Mili et al., 2002, p. 1). It is not like reusing raw materials that can be easily quantified in terms of their market value.

However, in order to measure the benefits of reuse, there must be a quantifiable measure that can be used to "assess the products and processes of software development" (GAO, 1993, p. 10). Despite the variety of assets available for reuse, software code is the most common asset referenced when describing reuse. This ignores the expertise and knowledge (intellectual property) gained by personnel who originally worked on the code. In some cases, simply providing software code without other assets can actually drive up costs (Mili et al., 2002, p. 7).

Other aspects of software code that make prediction of savings difficult is the programming language in which it was written and whether the amount of reused code is more or less than the amount of code that would have actually been required for the system. Software development is still an evolving discipline and there is no dominant programming language used across the industry. The variety of programming languages can have problematic effects on a development effort. The AN/BSY-2 program

experienced this when it attempted to use a type of programming language that was relatively new (GAO, 1989, p. 2). This required additional training costs to support the development effort.

Another shortcoming of expressing reuse benefits in terms of code is the variety of solutions for a problem available. A simple problem can be solved in a number of ways using varying amounts of software code in any number of languages. An internet challenge to find different ways to generate the lyrics of the song "99 Bottles of Beer" has generated over 1270 variations, all of which are valid (Schade, 2009).

## C. SOFTWARE REUSE COSTS AND BENEFITS

Several software cost models have been developed in an effort to provide cost estimators and financial managers a vehicle to determine the projected costs associated with software development. The intent of this research is not to describe or assess each model. The COCOMO II model from Boehm et al., (2000) is chosen, along with input from various cost analysts within the Air Force and Navy to provide a background of the factors that contribute to software development costs and explain how software reuse can affect the cost of a system.

It is important to understand that there are shortcomings with any cost model. The researcher was not able to analyze the data used to create the model, nor is that within the scope of this research. The purpose of this section is to illustrate that there is no simple or widely used metric for determining reuse success.

Because software development is labor intensive and expensive (Karlsson, 1995, p. 9), the primary cost driver in the development of a software system is the effort of developers. The ultimate goal of a cost model is to determine the amount of effort required to complete a software project and assign a cost to this effort. Boehm et al., (2000), express the effort in terms of "person-months" (p. 29). This number can then be converted into some form of compensation for the developers based upon site-specific information. The quantity of person-months required for the completion of a project is based primarily upon the size of the project and the productivity of the individuals

working on the project. COCOMO II uses “source lines of code (SLOC)” (Boehm et al., 2000, p. 29) as the basic measurement of size. This is consistent with how Navy cost estimators determine the size of a software project. Productivity can be estimated using historical data (Boehm et al., 2000) or can be determined from site-specific data based on the individuals working on a project. Boehm et al., (2000) expresses this number in “Person Months per Thousand Source Lines of Code” (p. 29).

In addition to these factors, Boehm et al., (2000) use multipliers that account for additional factors that can have an effect on the effort required to complete a project and scale factors that “account for the relative economies or diseconomies of scale for software projects of different sizes” (p. 30). Scale factors have numerical values associated with subjective ratings based on factors such as “Team Cohesion, Process Maturity and Precedentedness” (Boehm et al., 2000, pp. 33–34). Effort multipliers also assign numerical values based on subjective factors such as “Product Complexity, Required Software Reliability, and Personnel Experience” (Boehm et al., 2000, pp. 41–42). Descriptions of these and other multipliers can be found in Boehm et al., (2000) and Cummings, Gallo, Johnson, Marsh-Jones, and von Kuegelgen (1998). Use of multipliers such as those listed illustrates the level of subjectivity involved in accurately estimating the cost of software development. Thus, the effort required is affected by the estimated size of a software program, the productivity of the developers, factors accounting for economies or diseconomies of scale, and various multipliers that can directly affect effort.

Not mentioned yet in the discussion of software estimation factors is the effect of reuse on the cost of a project. Boehm et al., (2000) explain that “code taken from another source contributes to a product’s effective size” (p. 19). Two types of reused code are accounted for in the model: preexisting code that can be plugged directly into a product and preexisting code that must be modified prior to being reused. The categories of reusable software code are explained later, but it is important to understand how reused software relates to cost prior to describing the forms that reused software can take. To estimate the effective size of reused software code, Boehm et al., (2000) use a factor

called “equivalent source lines of code” (p. 19). This value is then added to the estimated source lines of code for a project. Converting reused code into equivalent source lines of code is necessary due to the non-linear effects of reuse.

Non-linear reuse effects originate due to the two types of software reuse efforts mentioned. Based on a study by Selby (as cited in Boehm et al., 2000) on the NASA Software Engineering Laboratory, two primary effects of the non-linear effects of reuse are evident. The first is that the effort associated with reusing software code does not start at zero (Boehm et al., 2000, p. 20). Selby found that the cost to reuse a line of code verbatim with no modification is equivalent to five percent of the cost of developing a new line of code (Boehm et al., 2000, p. 20). Software cost estimators at NAVAIR provide a variety of ranges. Depending on the contractor, the cost of an unmodified reused line of code can range from one percent to 25 percent of the cost to develop a new line of code. These numbers fall within the average determined by Selby, but certainly illustrate the uncertainty involved in the cost estimation process.

Five percent of the cost of a new line of code applies if the code being reused does not have to be modified. The second non-linear effect results from code that must be modified. Selby found that as more code must be modified, the costs of modification increase disproportionately to the amount of code being modified (Boehm et al., 2000, pp. 20–21). Boehm et al., (2000) attribute these increases to the cost of “understanding the software to be modified, and the relative cost of checking module interfaces” (p. 21). Again, there is no consensus on the relative cost of modifying reused code, but there is a disproportionate cost of having to modify lines of code. From Boehm et al., (2000) and cost estimators, this can typically range from 30 to 60 percent of the cost of developing a new line of code. Above a certain percentage of modification it is likely that the reused code will be more expensive to use than simply developing the code from scratch (Boehm et al., 2000, pp. 20–21). Within the cost model this value equates to approximately 25 percent. Therefore, if more than 25 percent of the code must be modified, then it may be more cost effective to simply develop the code from scratch.

To capture these non-linear effects, Boehm et al., (2000) again use multipliers to provide an estimate of the equivalent size of the project and the effort required for completion. These multipliers also assign quantitative values to subjective factors such as "Programmer Unfamiliarity, Software Understanding, and Assessment and Assimilation" (pp. 23–24).

Full descriptions of the multipliers can be found in the text. These further illustrate the uncertainty and subjective nature of quantifying the costs of developing software programs and products.

In 2004, the Department of Defense began requiring contractors to provide forms known as Software Resource Data Reports (SRDRs) at various stages of development for Acquisition Category (ACAT) I and IA programs (Software Resource Data Report [SRDR], 2005). In addition to programming language and development effort data, these SRDRs require contractors to provide three categories of software code: new code to be developed and delivered, modified code to be developed and delivered, and unmodified, reused code to be developed and delivered. The data contained within these SRDRs combined with the costs associated with the program can then be used to develop a more defense oriented model that incorporates reuse. Currently, different organizations rely on different models to predict the cost of software systems.

### 1. Types of Reuse

Now that the potential effect on costs has been discussed, it is important to understand the different forms that reusable software code can take. Prior to reusing any type of software code, an analysis is conducted to determine if and how it can be integrated into a system and whether it will be cost effective. It is beyond the scope of this research to describe all of the types and scope of the analysis, but they can be found in the literature (Karlsson, 1995; Mili et al., 2002; Prieto-Diaz and Freeman, 1987). In general, Prieto-Diaz and Freeman (1987, p. 6), states that code reuse involves three general steps.

- Accessing the code
- Understanding the code
- Adapting the code

This list is not inclusive of all situations involving the development of a software product, but provides background on the different types of reuse and the effort associated with their use.

#### *a. Black-box Reuse*

Black-box reuse is a software asset that can be integrated "verbatim" into a host system without any modification (Mili et al., 2002, p. 18). The ability to incorporate black-box software assets into a development program is the most desirable type of reuse due to the limited effect on effort. Users of black-box assets need only a limited understanding of the asset in order to facilitate its reuse and do not require significant knowledge of its design (Mili et al., 2002, p. 18).

#### *b. White-box Reuse*

White-box assets require analysis and modification before they can be integrated into a new system (Mili, et al., 2002, p. 18). Assets classified as white-box require the user to understand additional details when considering the asset for reuse, and thus entail more effort. This additional effort can be accompanied by increased cost and production delays. As discussed in the non-linear effects of software reuse, the total amount of additional effort is affected significantly by the amount of code that must be modified prior to reuse. Depending on the amount of modification necessary, it may be more cost-effective to simply develop the required software code from scratch.

Although not as beneficial to the organization, this is the most common type of reuse. An analysis of the SRDR database by NAVAIR cost analysts found that reused code is almost always accompanied by new or modified code. As discussed earlier, modifying code can incur a wide range of costs

Another interesting aspect of this type of reuse involves the uncertainty that can be involved even when the code has been modified. Analysis of 39 computer software configuration items by NAVAIR cost analysts revealed that 18 of the programs ended up with less reused lines of code than planned and 21 had the same or more. Further, of the total 39, 27 resulted in reused code being a smaller percentage of the total delivered code than was originally planned (M. Popp, personal communication, January 28, 2009). This illustrates that even if the amount of software reused by a program increases, the required code to modify and integrate it into a system may grow at a faster rate.

White-box reuse can result in software development savings. The literature provides details on the analysis required and the processes that should be in place in order to maximize these savings. Additionally, there are development processes available to mitigate the amount of modification necessary to effectively use existing software code. Ultimately, although there is a potential for savings from white-box reuse, without the proper analysis these savings can quickly evaporate as more code must be developed for modification and integration into a system.

#### *c. Commercial-off-the-Shelf (COTS)*

Boehm et al., (2000) describe COTS as "pre-built commercially available software components" (p. 237). Mili et al., (2002, p. 566) classify a component as commercial-off-the-shelf if it meets the following general characteristics.

- It is sold, leased, or licensed to the general public.
- Buyers, lessees, and licensees have no access to the source code.
- It is offered by a vendor that created it and is responsible for its upgrades.
- It is available in multiple identical copies on the market.

Use of COTS software products can be beneficial, but like white-box components, COTS typically requires modification prior to reuse in a new system. The new code required to integrate COTS components into a larger system is referred to as

"glue code" (Boehm et al., 2000, p. 249). As with other methods of reuse, there are arguments for and against the use of COTS components in systems. One of the main arguments for using COTS components is decreased development time from using "existing, market-proven, vendor-supported products" (Boehm et al., 2000, p. 237). Reducing development time can result in reducing the overall development cost. Boehm et al., (2000) also identify several risks involved with the use of COTS.

> Traditional costs associated with new software development such as the cost of requirements definition, design, code, test, and software maintenance. Additionally, the cost of licensing and redistribution right, royalties, effort needed to understand the COTS software, pre-integration assessment and evaluation, post-integration certification of compliance with mission critical or safety critical requirements, indemnification against faults or damage caused by vendor supplied components, and costs incurred because of incompatibilities with other needed software and/or hardware. (p. 238)

These risks can be compounded if COTS source code is not available to developers. However, the literature also expresses that if these risks can be effectively managed and evaluated prior to development of a product, COTS integration can be successful in driving down development time and costs.

Ultimately, the use of COTS is similar to white-box reuse in that it requires additional code to integrate the code into a new system. The difference is that this code comes in the form of "glue code" as opposed to modification of the original. As stated, COTS components can be more difficult to actually modify, therefore the motivation to reuse COTS is to find components that can be integrated into a system with as little extra code as possible. Again, like white-box, the amount of glue code depends upon the components being reused.

Despite the potential challenges associated with COTS, the ultimate benefits can be significant. In the development of the Virginia Class submarine command, control, communications, information (C3I) system, Lockheed Martin utilized COTS in 76 percent of the software and 78 percent of the hardware (Lockheed Martin, 2003). They attributed this to a reduction in development costs of five to one over

previous submarine combat systems (Lockheed Martin, 2003). COTS components do have advantages, but it is important to understand the associated risks and potential for additional licensing costs when using them in system development.

During times of rapid technological change, it is difficult for one firm or organization to develop and take advantage of all advances in technology. COTS products provide the opportunity to leverage the development efforts of many organizations in a variety of fields to provide greater capabilities.

### 2. Planned Reuse

Now that the costs associated with reuse and the various types of reuse have been described, the methods of incorporating this reuse into an organization's software development strategy are discussed. Planned software reuse is described in other ways within the literature. Karlsson (1995) refers to planned reuse as development for reuse. This distinction is important from a development and cost avoidance perspective. As identified by Poulin et al., (1993),

> Planned reuse starts early in the software life cycle and involves a thorough requirements study and domain analysis of the problem area. By doing this additional planning and domain analysis, organizations identify the factors that normally change in software. Examples are hardware or system software; user, mission, or installation; and function or performance. (pp. 573–574)

Domain analysis of software components is essential to the correct classification of components for future reuse. Mili et al., (2002) define a domain as an "area of knowledge or activity characterized by a family of related systems. A domain can also be defined by the common managed features that satisfy a specific market or mission" (p. 125). Methods of conducting domain analysis are outside the scope of this research.

Planned reuse is described first due to the wide acceptance in the literature that it results in greater "cost and productivity benefits" (Poulin et al., 1993, p. 575) than opportunistic reuse. These benefits hinge on the potential for future application of a component. Planned reuse requires that components are developed in a manner that allows for their future use with as little additional effort as possible. This requires that

components are developed generically and with enough documentation to enable future users to understand the details and functions of the component quickly and easily. Because software development costs are labor intensive, the additional effort to design components in this manner also incurs additional costs. Mili et al., (2002) state that on average, the cost to develop a software asset for reuse is 1.5 times greater than the cost to develop it for a specific application. The additional cost to develop therefore requires that there be sufficient future applications where the savings derived from reusing a component will offset the additional up-front costs necessary to develop a component for reuse.

Poulin et al., (1993) describe the savings from planned reuse as "reuse cost avoidance" (p. 583). Reuse cost avoidance comes from the difference between the combined savings from reusing components and the additional cost required to develop a component for reuse. Cost avoidance is classified into two components, development costs avoidance and service cost avoidance (Poulin et al., 1993). Development cost avoidance is derived from the reduction in overall effort required to develop a new software system from components initially developed for future use (Poulin et al., 1993). Service cost avoidance refers to the savings that can be achieved by not having to fix errors in code that has been reused (Poulin et al., 1993). The assumption here is that reused code will have fewer errors.

The following are examples of methods used in an effort to foster reuse within an organization.

#### *a. Software Product Lines*

In an attempt to take advantage of the greater benefits offered by software development through programs of planned reuse, organizations have developed structured business plans. One such plan is the Software Product Line (SPL) developed by Carnegie Mellon's Software Engineering Institute (SEI). The SEI website contains significant guidance concerning how to develop, maintain, and benefit from a product line approach to software development. The SEI defines a software product line as "set of

software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" (Software Engineering Institute [SEI], 2009). In a product line, core assets are developed and then other individual products are built according to a "pre-defined production plan" (Bergey, Fisher, Gallagher, Jones, and Northrop, 2000, p. 3).

Bergey et al., (2000) describe the fielding of a product line as "core asset development or acquisition and product development or acquisition using core assets" (p. 5). Also important to the concept is that the core assets and products do not have to be developed separately or in a specific order. Figure 1 (Clements et al., 2006) illustrates that core assets can come from products already in existence or from products developed using core assets (Bergey et al., 2000, p. 5). Additionally, core assets can be developed or procured first and then used to produce software that is not fully developed. The value of the core assets is based upon the number of products that can be developed using them (Bergey et al., 2000, p. 6). This value is not set and can vary depending upon the organization and the nature of the business.

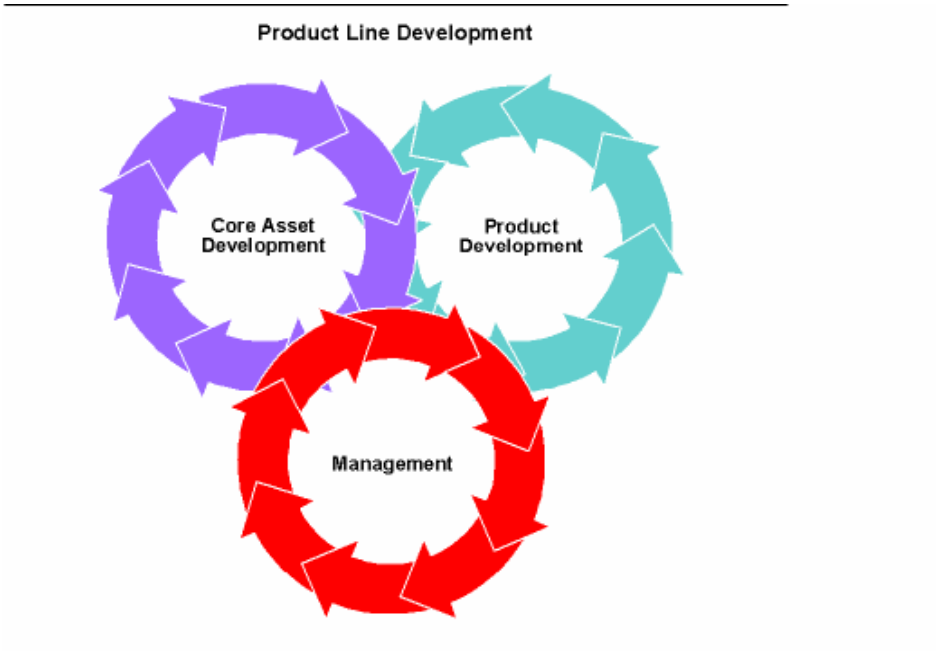


Figure 1. Software Product Line Essential Activities (From Clements et al., 2006)

There are numerous case studies, some specific to the Department of Defense and ASW software development that outline the process used to assess the benefits of a Software Product Line to the Department of Defense. Cohen, Dunn, and Soule (2002) use the Naval Undersea Warfare Center (NUWC) as the basis for one of the case studies. The results of the study indicate that NUWC was able to save $15 million from an investment of $3.5 million (Cohen et al., 2002, p. 35). NUWC was also able to reduce the development time from years to months for products developed within their Rangeware software product line (Cohen et al., 2002, p. 36).

In the lessons learned section, the study acknowledges that NUWC is the supplier and the customers were well defined prior to the implementation of the program. It further acknowledges that the model is similar to a contractor "supplying products built from assets" (Cohen et al., 2002, p. 38). The risk for the Department of Defense is that this may place heavy reliance on one contractor. Additionally for large system acquisitions, this may be beyond the ability of one contractor. The benefits of using a Software Product Line have been well documented. The Software Engineering Institute's website contains numerous case studies detailing these benefits.

Although the case studies presented illustrate the benefits in cost, development time, and quality of adopting a software product line approach to software development, there is also acknowledgement of the difficulties in adopting the approach within the Department of Defense acquisition environment. Due to downsizing of the acquisition workforce, the Department of Defense relies more on acquisition of "software intensive systems" rather than "in-house" development (Bergey et al., 2000, p. 1). For large Department of Defense programs, adopting the product line practice requires suppliers or contractors to develop core assets and another group of contractors or suppliers to then reuse these components (Bergey et al., 2000, p. 11). Taking advantage of such an approach requires a change in the acquisition culture for both contractors and the Department of Defense. Developers must have access to core assets and be willing to

use them in their own system development. Additionally, Department of Defense programs must be willing and have the vision to develop assets for future reuse (Bergey et al., 2000, p. 11).

### *b. Software Libraries*

Even when software components are developed for reuse, there must still be an effective system in place to minimize the non-linear reuse effect associated with retrieval. Although five percent of the cost of a new line of code is small, this cost can increase based on the difficulty in finding a component. Organizations have turned to the use of software reuse libraries in an effort to provide a common storage area for assets developed within an organization.

According to Mili et al., (2002), a useful reuse library must have a method of storing and retrieving components that improves the ability of developers to access and characterize them for reuse. There are also risks associated with libraries. According to Nelson (2007) "without active management and processes that encourage reuse, asset repositories will simply become dumping grounds for large numbers of files" (p. 28). Therefore, there are also costs associated with establishing and maintaining a successful reuse library. There must be effective storage and retrieval methods, as well as stringent evaluation of components. If a component is too difficult to find or retrieved components have errors, developers may determine that new design is a more effective method of completing a task.

An example of a software library is the Oceanographic and Atmospheric Master Library (OAML) operated by the Commander, Naval Meteorology and Oceanography Command (CNMOC). It is an example of a Navy software library of "models, algorithms, and databases that describe the ocean and atmosphere. It is updated and maintained by CNMOC and currently contains 63 products composed of 20 databases, 30 models, and 13 algorithms. The software products are "machine independent" and easily ported between systems (Naval Meterology and Oceanography Command [CNMOC], 2009).

Data collection for the library comes from a variety of sources including Navy labs, universities, contractors, private firms, and international data acquisition (CNMOC, 2009). Requirements can come from fleet needs, advances in technology, or from individual navy programs that have a need for the software updates. Once the requirements are defined and the data collected, the potential solutions are developed, verified, and validated prior to being accepted into the OAML database. Following acceptance, the new software products are distributed to the necessary fleet elements and integrated into new or existing systems. The products developed by OAML are designed for reuse in other systems and have been used extensively.

The benefits associated with the products developed by the OAML are unique from private software libraries in that they are developed to fulfill a necessary capability for operational units (CNMOC, 2009). The benefit for the Navy is that the assets developed for the database only have to be developed once. Additionally, because the assets are maintained and updated by a single organization, it is more efficient.

One aspect of setting up and maintaining a library is to determine the metrics used for evaluating its success. Because the Department of Defense is an acquirer of software systems, this can be difficult to determine. Mili et al., (2002) provide several metrics that can be used.

- Number of Accesses to the Library
- Number of Retrievals from the Library
- Library Efficiency

The number of accesses to the library is simply the number of times that the library is accessed in a given time period. This may be an indication of the perceived quality of the library (Mili et al., 2002, p. 490). However, it may not be indicative of actual uses of components.

The number of retrievals from the library refers to the actual assets extracted for reuse. This could be an indication of the actual quality of the assets available (Mili et al., 2002, p. 490).

Library efficiency is the "ratio of the number of reused assets that have been extracted from the library per unit of time over the total size of the library (Mili et al., 2002, p. 490). This gives an indication of the actual use compared to the size.

The importance of maintaining metrics is to ensure that the cost of placing assets into a library and subsequently maintaining the reuse library do not exceed the benefits. Ultimately, if a library is not being accessed or used with any measurable results, it should be removed from service.

### *c. Department of Defense Communities of Interest (COI)*

Department of Defense Chief Information Officer [DoD CIO] (2007) defines a Community of Interest (COI) as a "collaborative group of users who must exchange information in pursuit of their shared goals, interests, missions, or business processes and who therefore must have shared vocabulary for the information they exchange" (p. 3). A COI is designed to aid in sharing data between organizations that may require their use. This can include organizations within the Department of Defense and other agencies such as the Department of Homeland Security (DoD CIO, 2007). The intent is that the COI have an impact on acquisition. Overall, a COI would be established to share information between "known and unanticipated users" (DoD CIO, 2007, p. 4).

Chartering of a COI could provide greater opportunities for software reuse among the organizations involved. A COI could facilitate the standardization of data and ensure that the other members have a greater understanding of the assets and resources that are already available. Although such a level of sharing would be beneficial, getting the cooperation of all stakeholders such as program offices, end users, and developers cannot be viewed as a simple task. In addition, no extra funding is provided for the establishment of a COI (DoD CIO, 2007, p. 5). Participating organizations must plan their budgets to support COIs. Therefore, participation will have to be considered beneficial at all levels of an organization prior to obtaining funding for such an effort.

The potential for developing a software reuse program within the construct of a COI is available, but it will require some level of analysis to determine whether it is a worthwhile effort.

### 3. Opportunistic Reuse

Karlsson (1995) refers to opportunistic reuse as software development with reuse. Poulin et al., (1993) describe it as code recovery. Where planned reuse involves an organizational strategy and initial investment in developing components for reuse, opportunistic reuse simply takes advantage of assets such as software code that are already in existence. Poulin et al., (1993) consider opportunistic reuse as an inferior strategy to planned reuse and do not believe that the same degree of benefits can be achieved through opportunistic reuse. Opportunistic reuse can be more susceptible to the non-linear reuse effects discussed previously. A component not developed for reuse may not have all of the required documentation to make it cost effective to reuse.

An example of opportunistic reuse related to Anti-Submarine Warfare (ASW) comes from a software developer and analyst at the Naval Center for Cost Analysis that worked on both the Mk-118 Fire Control System for the Trident submarines and its predecessor, the Mk-117 Fire Control System, developed for fast attack submarines. While working on the Mk-118 project, the developer recommended “bringing over” or reusing applicable software from the Mk-117 (S. Oxman, personal communication, February 24, 2009). In doing so, the project was able to save both time and resources. One could investigate further and attempt to determine how much software code was reused, how long it took to integrate, and what the overall cost avoidance was in this instance. However, the basic fact is that this action saved both time and money. Additionally, there could be an evaluation of the processes in place that fostered the reuse, but in this case the reuse was conducted based on experience and an understanding of the system being developed. The reuse in this case is considered opportunistic because there was no formal plan for reuse. One of the managers was familiar with previous projects and was able to use that knowledge to benefit another project.

Despite the success achieved in this case, it is not the most efficient method of reusing software. Recognizing that if that particular developer had not been experienced in both projects, that particular instance of reuse may not have occurred and the project may not have been as successful. For this and other reasons, there has been a movement to formalize processes associated with the reuse of software components.

### 4. Organizational Issues

Despite the benefits that can be obtained from reusing software, some organizations are still resistant to incorporating widespread software reuse. Although organizations are motivated by financial results there are still other factors that contribute to the decision. Adopting widespread software reuse in companies requires a change in operational procedure (Mili et al., 2002, p. 7). Such a decision can affect all levels of an organization (Haefliger, von Krogh, and Spaeth, 2008, p. 181; GAO, 1993, p. 10). In some cases, the organizational obstacles can be more difficult to overcome than the technical issues (Haefliger et al., 2008, p. 181). Program managers, driven by external factors such as funding and schedule (GAO, 1993, p. 10) may not be willing to allocate additional time and resources to "develop reusable software components" (GAO, 1993, p. 10). A reluctance of developers to use components developed by someone else can also be a barrier to reuse (GAO, 1993, p. 10).

The Navy is not immune to these types of organizational issues. Although the Navy has recognized commonalities within the ASW mission area and the potential for reuse, the culture of the acquisition environment presents difficulties (Clements et al., 2006, p. 17). Just as the ASW mission is spread across multiple communities, the acquisition of ASW systems is divided among several program executive offices (PEO) (Clements et al., 2006, p. 12).

- PEO Subs
- PEO Ships
- PEO Littoral and Mine Warfare
- PEO Integrated Warfare Systems
- PEO Carriers

These organizations are further divided into program management offices that may deal more directly with specific ASW software applications. An example is PMS 401 in the PEO Subs. Its purpose is to oversee the development of "submarine-unique sonar systems" (Clements et al., 2006, p. 13). Clements et al., (2006) describe two challenges faced in trying to foster this development.

> Two particularly significant challenges to reuse included accommodating
>
> 1) a mix of management organizations responsible for fielding ASW systems. Five PEOs from across the various Navy communities share responsibility for ASW. Each oversees multiple contractors and labs under individual program managers. In addition, the Office of Naval Research and independent research labs also develop ASW software.
>
> 2) the varying nature of ASW within and across communities. The operational tempo of the submarine differs from that of the surface ship community. The air ASW mission has different operational concerns and is split between fixed-wing and helicopter ASW platforms. While the basics of ASW are similar across these communities, each perceived their requirements as being sufficiently unique that they could not develop their systems using common software. (p. 17)

Organizational constraints can have a significant effect on software reuse. Reuse efforts can require significant investment and the benefits may not be immediately achieved. Even within the Navy, taking advantage of software reuse between program offices requires coordination and a willingness to accept potentially higher initial costs for the achievement of future benefits.

In an effort to enhance this coordination among program offices that require ASW software, an ASW Community of Interest (COI) was chartered in 2007. The purpose of a COI within the Department of Defense is to "foster improved interoperability among National Security Systems" (Program Executive Office—Integrated Warfare Systems [PEO IWS], 2007, p. 1). The definition and role of a COI is described under Section B.2 of this chapter.

The ASW COI is described in its charter as an "institutional, cross-functional" community (PEO IWS, 2007, p. 2). It is divided into two groups, the users group and the developers group. The developers group is responsible for the acquisition and engineering aspect of the COI. This includes "coordinated software development and reuse" (PEO IWS, 2007, p. 2). Its goal is to provide a means of combining the efforts of the various organizations that develop and use ASW software. This will foster increased opportunities to develop products that have a greater potential for reuse. The following program offices are represented in the developers group of the COI (PEO IWS, 2007, p. 2).

- PEO Integrated Warfare Systems (IWS)
- PEO Air ASW, Assault & Special Mission Programs (A)
- PEO Submarines
- PEO Littoral and Mine Warfare (LMW)
- PEO Command, Control, Communications and Information (C4I)
- Office of Naval Research (ONR)
- Warfare Centers with ASW responsibilities (no additional description provided)

In addition, the COI includes an ASW Executive Steering Group (ESG) that will handle the "day-to-day group activities" (PEO IWS, 2007, p. 2). Success of the COI will be determined using defined metrics such as "time-to-market, software reuse, and the number of entries into the DoD Metadata Registry" (PEO IWS, 2007, p. 4). Additional

metrics such as “operational performance, risk, and return on investment (ROI)” will be incorporated when methodologies for measurement are established (PEO IWS, 2007, p. 4). Data reflecting the success of these metrics was not available at the time of this research.

Although still in the early stages, the establishment of an ASW COI illustrates a desire to overcome the organizational issues that can hinder a reuse program. Gaining the support of the top levels of management within the listed program offices could substantially increase opportunities for planned reuse of ASW software. However, it will also have to provide proof that such an effort is beneficial to all of the listed stakeholders.

In the development of ASW software, the Navy relies heavily on the expertise of system integrators and contractors. Within many of these organizations, the opportunity for reuse is substantial. At Lockheed Martin Maritime Systems and Sensors in Manassas, Virginia, the presence of several programs that rely on ASW software provides an opportunity for sharing of software. Labs used to develop and integrate software for the submarine, surface, air, and surveillance communities are in close proximity and allow for collaboration between developers. This may not correlate directly to cost savings for the Navy, but seems to be the state of the practice.

Reuse within a single organization is not the only method currently available for reuse. Examples also exist of sharing between contractors mandated by the program office that maintains control over the developed software. One example is from PMS 425 within PEO Subs. They orchestrated the supply of several million lines of code to the DDG-1000 program (R. Jackson, personal communication, March 23, 2009). However, the status of this reuse was not available at the time of this research.

### 5. Benefits of Reuse

The actual costs associated with software reuse are largely quantifiable. Conversely, benefits associated with the reuse of software components are not all as quantifiable. Benefits can be both internal and external to an organization and have been well documented in the literature. Generally, they include cost avoidance, improved

quality, and reduced development times (Mili et al., 2002, p. 6; Poulin et al., 1993, pp. 587–591). These benefits can also be applied to the stated goals of Department of Defense software acquisitions.

### *a. Cost Avoidance*

Cost avoidance in the Department of Defense is "a reduction or elimination of some future resource requirement" (American Society of Military Comptrollers [ASMC], 2008, p. 2.2.31). The concept of reuse cost avoidance was introduced by Poulin et al., (1993). The most basic benefit is the actual cost savings from reusing software. This is the most quantifiable aspect of benefits. If the costs of reuse are lower than the costs of developing new software, then there is clearly a benefit to the organization (Poulin et al., 1993, pp. 589–590) in reusing the software and the applicable components should be reused.

More detailed analysis of this approach can be found in Poulin et al., (1993). In general, the cost avoidance is determined by finding the difference between the additional cost in developing software for reuse and savings from reusing the software in the design of future components and reduced service costs from using software that is of greater quality.

The equation describing this concept is illustrated below.

RCA = ADC—DCA—SCA

Where:

RCA = Reuse Cost Avoidance

ADC = Additional Development Cost

DCA = Development Cost Avoidance

SCA = Service Cost Avoidance

Fundamentally, this equation illustrates that the primary savings from reusing software can be found in the reduction of future development costs and the costs associated with maintenance and support. An organization that is successful in reusing software should see reductions in costs such as RDT&E and maintenance. If tangible

benefits are not achieved, then perhaps the organization used incorrect assumptions or the expected opportunities for reuse were not available.

In relation to the acquisition of Navy weapons systems, calculating cost avoidance from software reuse is made more difficult by the lack of accurate data representing the quantity of reuse. Development costs associated with previous systems may have benefitted from the reuse of software, but meaningful data supporting this idea are not readily available. Additionally, the shortcomings described previously in the use of source lines of code and cost estimation models will also increase the inaccuracy of the potential cost avoidance calculation.

#### *b. Quality*

Karlsson (1995) defines quality as "the ability of software to meet its requirements" (p. 6). Although broad, this definition encompasses what decision makers need to address when viewing software products. If the product fails to satisfy Karlsson's definition then it should not be considered. This can also refer to problems or bugs within the software. Using a software component multiple times increases the likelihood that any problems or issues with the software have been fixed. There are numerous definitions of what pertains to quality within the literature. Each organization should determine its own definition for a quality product within the context of their needs.

Improved quality for the Navy can have both tangible and intangible benefits. An example of a tangible benefit is a reduction in corrective maintenance that must be funded. GAO estimated spending to rework software systems at $8 billion, representing 40 percent of the amount of money spent on software systems that year (GAO, 2004, p. 1). Those numbers only represented money spent in research, development, test, and evaluation. Support costs for software systems typically make up a significant amount of the lifecycle costs, but were not included in the GAO estimate. Thus improving the quality of the final product can result in actual cost savings to the Navy.

### *c. Development Schedule*

By taking maximum advantage of components that can be reused, an organization can reduce the amount of time that it takes to develop a product. For private industry this carries the advantage of getting products into the market at a more rapid pace and saving money on labor costs (Mili, et al., 2002, p. 6). The Navy also stands to gain similar benefits from reducing the amount of time that it takes to develop software systems. Developing systems in a timelier manner can reduce overall production costs. As stated earlier, the primary cost driver in the development of software systems is labor. Reducing the amount of time that it takes to field a system will ultimately reduce labor related costs.

In private industry, the benefit is an ability to get products into the market faster, thus potentially enabling the company to sell more of the product. The Department of Defense is primarily focused on the acquisition of products, thus the benefit comes from reduced acquisition costs associated with development. Ensuring that a software system is developed on time allows integration into the final product more rapidly. This can help to ensure that the larger system is also delivered on time.

An intangible benefit of faster delivery is the introduction of capabilities to operators more rapidly. This can aid in mission accomplishment and give personnel an edge over adversaries. Although this is a benefit attributable to faster development time, it is not one that is easily quantified.

## 6. Summary

Software reuse is a complex issue that requires an understanding of the associated costs and benefits that can be achieved. There are different classifications of components depending on whether they simply can be plugged into a new system or whether they require additional modification and effort to reuse. Consideration must also be given to whether they were developed in-house or purchased as commercial-off-the-shelf. Each type carries with it costs that can affect the overall benefits achieved. Even when components are chosen for reuse, there must still be an organizational strategy on how

reuse will be accomplished. Several models are available to determine cost avoidance associated with reuse and development efforts that can be associated with reuse. Limited benefits can be achieved by simply reusing code that is available, but more successful examples of reuse require a process that is accepted and supported by all levels of an organization.

It is unclear how to classify the current methods employed for ASW software reuse. The state of the practice involves a reliance on contractors to leverage developed assets across communities. Reuse is a practice that has become inherent in their development processes. This has resulted in some successful efforts, but makes precise cost avoidance associated with reuse difficult to determine. PEO IWS 5 estimates that an APB developed for the submarine community may result in the following levels of reuse (C. Davis, personal communication, February 10, 2009):

- Submarine to submarine: 97%
- Submarine to surveillance: 40%
- Submarine to surface: 25%
- Submarine to air: 15%

There are also estimates that the Navy has saved over $500 million from avoiding the development of 4.5 million source lines of code (Clements et al., 2006, p. 29).

Clements et al., (2006) is an analysis conducted by the Software Engineering Institute that compares the Navy's APB process and subsequent management of ASW software to that of the Product Line approach. The report finds multiple differences between the APB process and a Software Product Line. However, it also recognized the efforts of the Navy in its management of ASW software and the positive results that it has been able to achieve thus far. This includes both economical and technical benefits.

THIS PAGE INTENTIONALLY LEFT BLANK

# III. METHODOLOGY

## A. GENERAL APPROACH

The following general approach was developed in order to determine whether the Navy has achieved cost savings from the reuse of anti-submarine warfare sonar and fire control software.

1. Determine the relationship between software reuse and the associated costs of developing and maintaining software intensive systems.
    a. Interviews with cost analysts, acquisition professionals, and software engineers
    b. Literature review
2. Determine the trend in the cost of software development related labor
    a. Select applicable employment statistics from the Bureau of Labor Statistics website
    b. Determine the wages for software related labor and normalize to 2009 dollars
3. Select sonar and fire control systems for analysis
    a. Normalize cost data to fiscal year 2008 dollars to provide a common baseline for comparison
    b. Segment costs potentially related to software reuse
4. Select research, development, test, and evaluation budget line items for analysis
    a. Normalize cost data to fiscal year 2009 dollars to provide a common baseline for comparison
    b. Segment cost data for analysis
5. Identify trends in the funding streams
6. Provide observations and recommendations for future analysis

## B. SOFTWARE DEVELOPMENT LABOR COSTS

As discussed in Chapter II, labor is the primary cost driver in the development of computer software. In order to facilitate a proper analysis of the costs associated with ASW software development, there must be an accompanying analysis of the trends associated with this primary cost driver.

The following three occupations were chosen due to their relationship to the development of computer software. All job descriptions were obtained from the Occupational Employment Statistics (OES) section of the Bureau of Labor Statistics (BLS) website (http://www.bls.gov).

- **Computer Programmers:** Convert project specifications and statements of problems and procedures to detailed logical flow charts for coding into computer language. Develop and write computer programs to store, locate, and retrieve specific documents, data, and information.
- **Computer Software Engineers—Applications:** Develop, create, and modify general computer applications software or specialized utility programs. Analyze user needs and develop software solutions. Design software or customize software for client use with the aim of optimizing operational efficiency. May analyze and design databases within an application area, working individually or coordinating database development as part of a team.
- **Computer Software Engineers—Systems:** Research, design, develop, and test operating systems-level software, compilers, and network distribution software for medical, industrial, military, communications, aerospace, business, scientific, and general computing applications. Set operational specifications and formulate and analyze software requirements. Apply principles and techniques of computer science, engineering, and mathematical analysis.

After the three occupations were selected, National Mean Hourly Wage data for 1999 to 2007 were collected from the OES section of the BLS website. The wage data on the site is given in then-year dollars and must be converted to a common year in order to facilitate comparison and identify trends. All wages were converted 2009 dollars using the Consumer Price Index (CPI) calculator, which is also located on the BLS website. The data were then plotted versus the corresponding year to identify trends in the cost of software development labor.

## C. SONAR AND FIRE CONTROL SYSTEMS SELECTED FOR ANALYSIS

The primary reason for choosing the following sonar and fire control systems is that they are recognized by PEO IWS 5—one the Navy program offices responsible for undersea systems - as systems that have reused ASW software. They were also chosen because of their levels of use within the submarine and surface communities. Although other systems are still in limited use within each community, the researcher considered it

important to analyze the equipment that is being maintained and updated for continued and future use. These systems are also being incrementally updated to take advantage of new technology. For the submarine community, the AN/BQQ-10 Sonar System (ARCI) and the AN/BYG-1 Submarine Combat Control System represent the baseline for the development of future common systems. The future common surface ASW system has not been identified, but the AN/SQQ-89 Surface Combat System represents a system that is being updated based on advances in hardware and software technology.

Following the identification of the systems, maintenance and training data were obtained through the Navy's Visibility and Maintenance Operating Support Costs (VAMOSC) database. Operating and Support Costs (O&S) were used for the following time frames:

- AN/BQQ-10: fiscal years 2001 to 2007
- AN/BYG-1: fiscal years 2004 to 2007
- AN/SQQ-89: fiscal years 2000 to 2007

The data were provided by VAMOSC in fiscal year 2008 dollars. Data for each system were then broken out by the following categories and descriptions. All descriptions were obtained from the VAMOSC website (www.navyvamosc.com):

- **Engineering and Technical Services (ETS):** Cost of engineering and technical services provided to support a shipboard system other than during intermediate or depot availability. These services are provided by Navy engineering activities or by contractors.
- **Software Support:** The cost of software maintenance and modification for embedded computer software.
- **Training NETPDTC** (Naval Education and Training Professional Development Technology Center)**:** Cost of C,D, F, G, and T formal course training for the ship's crew, officer and enlisted to enable them to operate and maintain a shipboard system.
- **Training Program Office:** The Program-Office funded costs of training personnel to operate and maintain a shipboard system.
- **Number of Systems:** The total number of systems installed/utilized by the Navy that are onboard ships that are in commission, as reported by program offices.

- **Personnel Trained—Program Office:** The total number of sailors trained during the fiscal year to operate and maintain a shipboard system.
- **Personnel Trained—NETPDTC:** The total number of sailors trained during the fiscal year to operate and maintain a shipboard system.

Descriptions concerning the NCCA's collection of the data are found in the Shipboard Systems User Manual located on the VAMOSC website. A brief summary is provided below.

ETS costs, software support costs, number of systems, training—program office, and personnel trained—program office are provided by the program offices to the NCCA. These costs are then compared to historical costs to identify deficiencies or anomalies (Naval Center for Cost Analysis [NCCA], 2009). If significant discrepancies are found, the provider is contacted for an explanation. Following any necessary clarifications, the data are added to the VAMOSC database for the applicable system (NCCA, 2009).

Training costs and personnel reported from the NETPDTC are subjected to a different process. The NETPDTC provides the number of graduates for each course identification number (CIN) associated with the learning center, the number of instruction days for each CIN, and the overall cost per instruction day of the learning center (NCCA, 2009). The NCCA then allocates the number of graduates and the costs of each course to specific shipboard systems (NCCA, 2009). The applicability of each course to specific shipboard systems is provided by the system's program office.

### 1. Relationship to Software Reuse

These costs were chosen due to their potential relationship to software reuse. However, costs or reduction in costs cannot be directly attributed to instances of software reuse by the chosen systems. The data used will simply provide evidence as to whether the current methods of development, which have been identified as taking advantage of software reuse, have been successful in lowering maintenance and training costs.

### *a. ETS and Software Support Costs*

As identified in previous chapters, instances of software reuse can result in higher quality final products. The systems provided to the Navy are finished products, so any quality gains should be realized in the development and testing phase. However, reuse during the stages of system design and production could result in a higher quality product being provided to the Navy. According to Clements et al., (2006), "a defect found in an asset on one platform is tracked throughout the fleet. Then, fixes repair that defect on all installed systems using the affected asset" (p. 29). Therefore lower support costs could be achieved through more rapid identification and isolation of faults on systems using similar assets.

### *b. Training*

Training costs were chosen due to the increase in commonality that can result from increased instances of reuse. In the literature, training typically refers to the individuals developing software assets and not the end users. The researcher acknowledges this and is exploring the possibility that as a specific warfare community achieves greater commonality of shipboard systems, which can come through software reuse, training costs may decrease.

## 2. Analysis

Once costs for each of the elements were separated, they were used to conduct the following analysis:

- Trend analysis of total engineering and technical support (ETS) and software support costs per unit per year
- Annual percentage change in ETS and software support costs per unit
- Trend analysis of total training costs per unit per year
- Trend analysis of training costs per individual trained
- Annual percentage change in training costs per unit
- Annual percentage change in training costs per individual trained

The goal of the analysis is to determine whether the trends indicate an increase or decrease in the O&S costs. These results could then be potentially linked to software reuse and provide evidence of successful reuse efforts.

### *a. Trend Analysis of ETS and Software Support Costs*

Trend analysis was conducted on a cost per unit per fiscal year basis. For this analysis, unit refers to the number of systems installed on U.S. Navy ships. The total ETS and software support costs from the VAMOSC database were summed and divided by the number of units of each system. This gave software related maintenance costs per unit. The purpose of this conversion is to provide a common baseline of funding.

These values were then plotted against the year in which costs were incurred. Costs per fiscal year are illustrated in graphical form in order to provide evidence of trends.

### *b. Trend Analysis of Training Costs*

The first trend analysis was done on a cost per unit per fiscal year basis. For this analysis, unit refers to the number of systems installed on U.S. Navy ships. The total training costs from the NETPDTC and program offices were summed and divided by the number of units of each system. This provided training costs per unit. These values were then plotted against the year in which the costs were incurred.

The second trend analysis was done on a cost per individual trained per fiscal year basis. The total training costs from the NETPDTC and program offices were summed and divided by the total number of personnel trained. This provided training costs per individual trained. These values were then plotted against the fiscal year in which the costs were incurred.

### *c. Annual Percentage Change*

The annual percentage change of each of the cost categories was calculated to provide additional evidence concerning either an increase or reduction in costs. Percentage changes were computed for ETS and software support costs per unit,

training costs per unit, and training costs per individual trained. The annual percentage change was calculated by subtracting previous fiscal year costs from the current year and dividing by the previous year.

$$\text{Annual Percentage Change} = \frac{\text{Current Year Costs—Prior Year Costs}}{\text{Prior Year Costs}}$$

### 3. Summary

This section provides additional details concerning the potential effect that software reuse has on maintenance and training costs for selected submarine and surface community ASW systems. The costs used in the analysis represent the total costs attributed to each system in a given fiscal year. They are not directly representative of costs or savings that can be attributed to reuse. What the analysis illustrates is whether training and support costs associated with system software have been increasing or decreasing on an annual basis.

## D. BUDGET LINE ITEMS SELECTED FOR ANALYSIS

The decision to use only R-1 budget line items from the RDT&E portion of the budget is based on the belief that a significant portion of the costs of a software system are incurred in the development and testing phase. In selecting the specific budget items, only programs related to anti-submarine warfare sonar and fire control software were chosen. Rationale for choosing the programs came from analysis of the Exhibit R-2 mission description and budget item justification and the project cost analysis in Exhibit R-3. Funding data selected for analysis are from fiscal years 2000 to 2008.

RDT&E costs are also separated into budget activities (BA). BAs are numbered from one to seven and correspond to different types of RDT&E. The Department of Defense Financial Management Regulation (DoD FMR) (2008) Volume 2B Chapter 5 provides a description of each BA. The only BA categories used in this research are BA 2, 3, 4, 5, and 7. Brief descriptions of BAs 2, 3, 4, 5, and 7 are provided below.

- **Budget Activity 2, Applied Research:** Applied research is systematic study to understand the means to meet a recognized and specific need. It is a systematic expansion and application of knowledge to develop useful materials, devices, and systems or methods. Applied research may translate promising basic research into solutions for broadly defined military needs, short of system development. The dominant characteristic is that applied research is directed toward general military solutions and determining their parameters.
- **Budget Activity 3, Advanced Technology Development:** This BA includes development of subsystems and component and efforts to integrate subsystems and components into system prototypes for field experiments and/or tests in a simulated environment. Advanced Technology Development demonstrates the general military utility or cost reduction potential of technology when applied to different types of military equipment or techniques. Projects in this category do not necessarily lead to subsequent development or procurement phases, but should have the goal of moving out of Science and Technology and into the acquisition process.
- **Budget Activity 4, Advanced Component Development and Prototypes (ACD&P):** The ACD&P phase includes system specific efforts that help expedite technology transition from the laboratory to operational use. Emphasis is on proving component and subsystem maturity prior to integration in major and complex systems.
- **Budget Activity 5, System Development and Demonstration (SDD):** SDD programs conduct engineering and manufacturing development tasks aimed at meeting validated requirements prior to full rate production. The SDD phase involves the development, integration, and demonstration of mature systems.
- **Budget Activity 7, Operational System Development:** This BA includes development efforts to upgrade systems that have been fielded or have received approval for full rate production.

### 1. Individual Budget Items Selected

The following process was utilized in determining the RDT&E budget items to be used in the analysis.

First, a search was conducted to identify budget line items related to the anti-submarine warfare mission. The search was conducted within the Department of the

Navy Budget located on the Assistant Secretary of the Navy (ASN) Financial Management & Comptroller (FM&C) website, http://www.finance.hq.navy.mil.

Potential items were then analyzed to determine if they were specifically related to the development of products in support of shipboard systems. This was done through analysis of the Exhibit R-2 budget item justification and mission description of each program. The goal of this analysis was to ensure that the funds allocated were being used in support of anti-submarine warfare sonar or fire control related systems.

Once the determination was made that the line item was directed towards anti-submarine warfare, the cost categories within Exhibit R-3, RDT&E project cost analysis, were used to determine the applicability to software development.

Not all cost categories specifically refer to software development. Nomenclature used within Exhibit R-3 varies. Only costs specifically identified as hardware were eliminated from consideration in the software cost analysis. Software related costs were then separated from the total costs of the program elements being analyzed. The following categories of costs were eliminated to differentiate between software and total costs:

- Primary Hardware Development
- Ancillary Hardware Development
- Tech Insertions (TI)
- Miscellaneous
- Award Fees
- Any costs considered classified and not explained within Exhibit R-2 or R-3

All other costs in Exhibit R-3 associated with product development, support, test and evaluation, and management are included.

When the software related costs were determined, they were recorded at their actual value and normalized to fiscal year 2009 dollars (FY09$) using the RDT&E (purchase) appropriation from the Naval Center for Cost Analysis (NCCA) inflation calculator (February 2009 edition).

Every effort was made to ensure that these costs represented actual values. This was done by using the most up to date data available. For example, fiscal year 2006 data were taken from the fiscal year 2009 budget, which was submitted in February 2008. RDT&E funding is available for obligation for a period of only two years. Thus all obligated fiscal year 2006 funds should be recorded in the fiscal year 2009 budget.

The researcher acknowledges that the costs associated with these programs may not correlate directly to actual reuse efforts. The purpose of using these data is to identify any trends in the RDT&E stages of anti-submarine warfare sonar and fire control software development. If processes in place to reuse anti-submarine warfare are successful, there should be a downward trend in the allocation of funding toward the development of such systems. However, even these trends may not exclusively represent successful reuse efforts.

A relationship may also exist between the funding profiles of each item. The items chosen are not inclusive of all efforts toward the development of anti-submarine warfare sonar and fire control software. They were chosen due to their direct relationship with developing sonar and fire control systems and because they have maintained a steady stream of resources for a significantly long period of time to facilitate analysis.

The following budget items were selected based on the reasons discussed above. They are listed by budget activity and program element. A brief description taken from the fiscal year 2009 Department of the Navy (DoN) Budget is also provided.

- **Budget Activity (BA)—4, Program Element 0603561N/Advanced Submarine System Development—Project Unit 0223/Sub Combat System Improvement (ADV):** This budget item provides funding for the development of the submarine community's Advanced Processing Builds (APBs) described in Chapter II. APBs are software only updates developed for integration into existing shipboard systems. They include both acoustic (sonar) and tactical (fire control) software development. According to Exhibit R-2a for fiscal year 2009, "APBs develop and demonstrate improvements to current and future sonar/combat control systems" (Department of the Navy Financial Management and Comptroller [DoN FM&C], 2008).

- **Budget Activity (BA)—4, Program Element 0603553N/Surface ASW—Project Unit 1704/Undersea Warfare:** According to the fiscal year 2009 Exhibit R-2 description, this budget item provides funding for "advanced development demonstration and validation of technology for potential surface sonar and combat system applications" (DoN FM&C, 2008). The goal is to identify technology that can be implemented into surface ASW sonar and fire control systems.

- **Budget Activity (BA)—5, Program Element 0604261N/Acoustic Search Sensors—Project Unit 0480/ASW Sensors & Processors:** The fiscal year 2009 Exhibit R-2 description explains that this budget item provides funding to "improve the mission effectiveness of airborne ASW platforms in cueing, search, localization, and track" (DoN FM&C, 2008). The goal is to provide solutions that better enable the detection, classification, and tracking of submarines.

- **Budget Activity (BA)—5, Program Element 0604503N/Submarine System Equipment Development—Project Unit 0219/Sub Sonar Improvement (ENG):** This budget item provides funding to develop and deliver updates to existing submarine sonar systems. Part of the funding is used to integrate the software upgrades developed by the APB process. The remainder is used for the development of hardware for the Technology Insertions (TI) described in Chapter II.

- **Budget Activity (BA)—5, Program Element 0604562N/Submarine Tactical Warfare System—Project Unit 0236/SSN Comb Cont Sys Imprvmnt (ENG):** This budget item provides funding to develop and deliver updates to existing submarine fire control systems, specifically the AN/BYG-1 combat control system. Part of the funding is used to integrate the software upgrades developed by the APB process. The remainder is used for the development of hardware for the Technology Insertions (TI) described in Chapter II

- **Budget Activity (BA)—7, Program Element 0205620N/Surface ASW Combat System Integration—Project Unit 0896/AN/SQQ-89 Modification and Budget Activity (BA)—7, Program Element 0205620N/Surface ASW Combat System Integration—Project Unit 1916/Surface ASW System Improvement:** These budget items provide funding to develop and integrate software upgrades for the AN/SQQ-89 system. According to the description, the two items will be combined in future budgets under project unit 1916. For this research the total costs of each are combined.

Following the selection of the budget items and the normalization of cost data to fiscal year 2009 dollars, analysis of the funding streams was conducted to determine the following:

- Trends in the RDT&E costs for each budget line item
- Annual percentage change in funding for each budget line item
- Trend in total costs for the six items selected

### 2. Relationship to Software Reuse

One of the potential benefits associated with software reuse is reduced development costs (GAO, 1993, p. 5). This can come in the form of shorter development time and greater productivity. Therefore, the use of RDT&E funding data is appropriate when trying to determine benefits associated with software reuse.

However, a lack of empirical data representing the extent of reuse within and between the various program elements is a significant shortcoming in the analysis. Any conclusions drawn from analysis of these data may not represent only the effect of software reuse. Other factors may have been instrumental in reductions or increases in the yearly expenditures associated with each budget item.

The data chosen are also not all inclusive of potential anti-submarine warfare related sonar and fire control costs and do not reflect all potential areas of software reuse. However, the budget items chosen represent elements that have received consistent funding and support software upgrades to existing systems.

### 3. Analysis

Following the determination and segregation of costs, trend analysis and computation of annual percentage change of RDT&E costs was conducted.

#### *a. Trend Analysis*

Trend analysis was conducted by plotting the RDT&E funding in fiscal year 2009 dollars for each budget line item against the fiscal year in which the funding

occurred. Once the funding stream was plotted, a linear trend line was drawn to determine the trend in spending over time. A downward trend in funding may be indicative of savings that can be attributed to the reuse of software within the program.

#### *b. Annual Percentage Change*

The average yearly change of each of the cost categories was calculated to provide additional evidence of either an increase or decrease in costs. Averages were computed separately for each RDT&E cost. The average percentage change was calculated by subtracting previous fiscal year costs from the current year and dividing by the previous year. An average of these percentages was then taken to determine the average annual percentage change.

$$\text{Annual Percentage Change} = \frac{\text{Current Year Costs—Prior Year Costs}}{\text{Prior Year Costs}}$$

### 4. Total Sonar and Fire Control Costs

One of the future endeavors in the ASW community is the implementation of an Anti-Submarine Warfare (ASW) Community of Interest (COI). The purpose of this section is to identify fiscal year 2008 and fiscal year 2009 RDT&E funding related to sonar and fire control systems that support the ASW mission. Costs are not segregated in this section and both hardware and software related costs are included. The purpose is to provide an indication of the amount of funding that is currently being allocated to ASW sonar and fire control systems within the RDT&E budget. Depending upon the level of funding, it may indicate whether such collaboration is worthwhile. No minimum or maximum threshold has been identified.

The R-1 budget items applicable to developing technology to support sonar and fire control systems were determined by reviewing Exhibit R-2 budget item justifications and mission descriptions. Only budget items that develop technology for ASW sonar and fire control systems were used.

Fiscal year 2008 costs were normalized to fiscal year 2009 dollars using the NCCA inflation calculator (February 2009 edition). The budget activity, program element, project title, and funding are displayed in table form in Chapter IV.

### 5. Summary

The costs used for analysis in this section were extracted from RDT&E budget data associated with ASW software development for the submarine, surface, air, and surveillance communities. Data were analyzed to determine any trends in the funding streams within each of the programs and the total funding related to developing sonar and fire control systems.

Based on anecdotal evidence concerning the reuse of ASW software, the expectation is that software related RDT&E costs will exhibit a downward trend. A downward trend in funding could be indicative of cost savings. However, it is important to note that any increases or decreases in funding could be influenced by other factors related to the budget process. This includes events such as congressional additions and reprogrammings. Therefore, any trends identified may not be directly related to software reuse.

Determination of total RDT&E funding for sonar and fire control related systems is meant to illustrate how much money is spent on these systems. In 2007, a charter was signed to establish the ASW COI. The information gathered from this analysis may provide evidence of the extent of the funding involved in the collaboration and allow for more informed decision making.

# IV. RESULTS FROM ANALYSIS OF THE DATA

This section presents the results of investigating cost savings associated with the reuse of anti-submarine warfare sonar and fire control software. The results explained below are representative of trend analysis conducted on labor, maintenance, training, and RDT&E costs associated with the development and sustainment of ASW related software.

## A. SOFTWARE RELATED WAGES

As stated in Chapter II, the primary cost driver in the development and support of software is labor. Figures 2, 3, and 4 illustrate the trend in the national mean hourly wages from 1999 to 2007 for Computer Programmers, Computer Software Engineers - Applications, and Computer Software Engineers—Systems Software, respectively. Table 1 illustrates the three areas of employment, as well as the national mean hourly wage associated with each profession from 1999 to 2007. Wages in 2009 dollars are shown in parentheses. Although the data do not exhibit a well-behaved linear trend, there is a general movement in the upward direction. The largest increases are from 1999 to 2000 and from 2003 to 2004. The largest decrease occurs from 2005 to 2006.

Overall, the wages for each occupation have increased from their 1999 levels. Specific reasons for the year-to-year changes are not within the scope of this research. The purpose of displaying these data is to illustrate that the cost of labor has increased over the time period of the analysis.

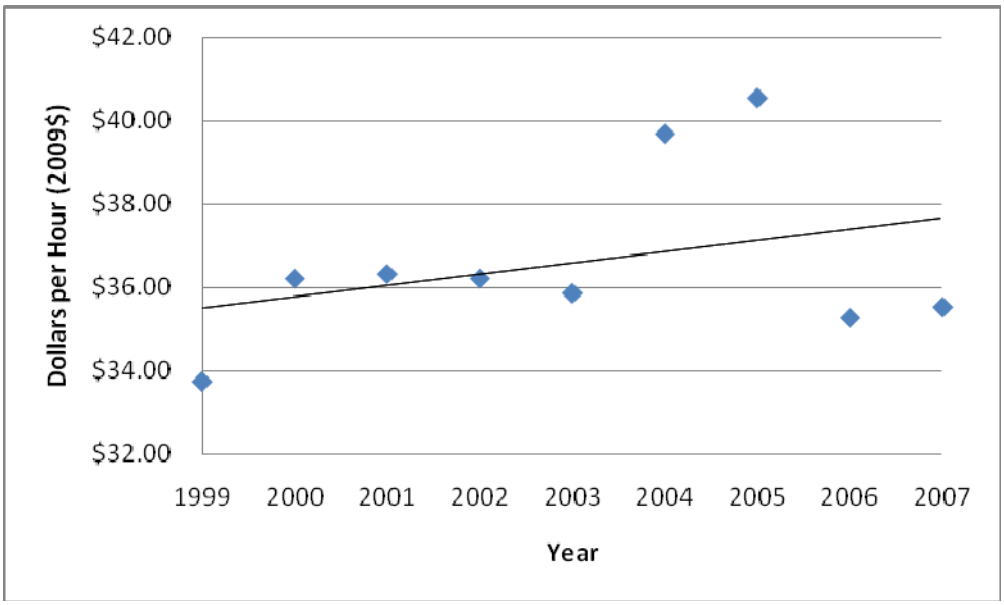


Figure 2. National Mean Hourly Wage for Computer Programmers from the Bureau of Labor Statistics Occupational Employment Statistics (Adjusted to 2009 dollars using the Consumer Price Index) vs. Year

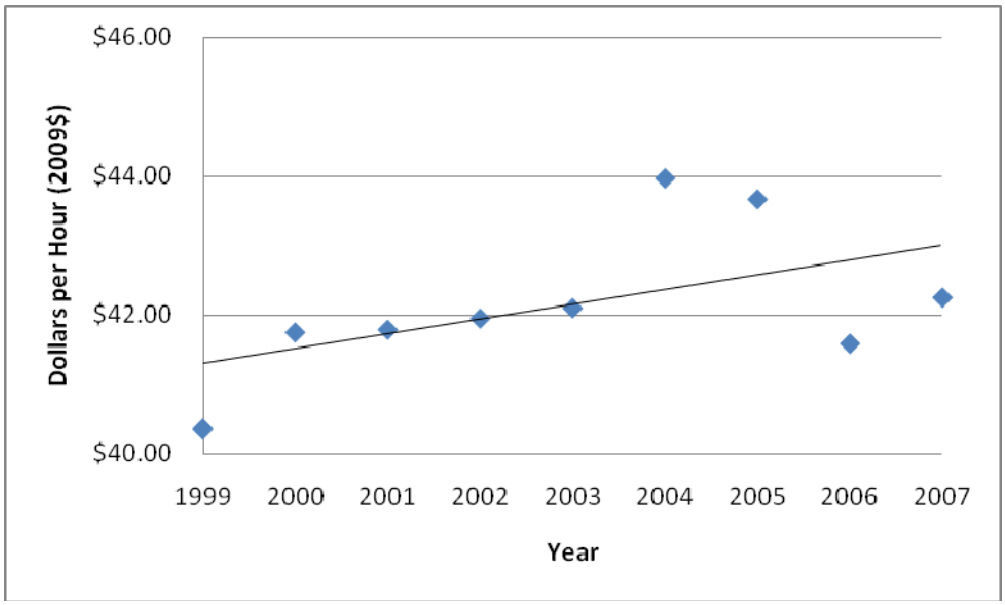


Figure 3. National Mean Hourly Wage for Computer Software Engineers - Applications from the Bureau of Labor Statistics Occupational Employment Statistics (Adjusted to 2009 dollars using the Consumer Price Index) vs. Year

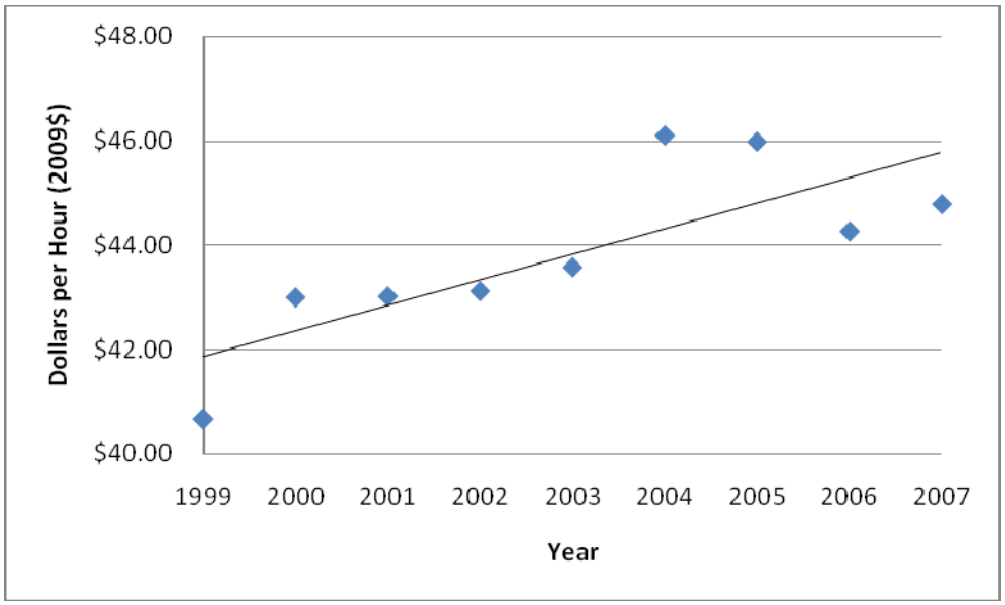


Figure 4. National Mean Hourly Wage for Computer Software Engineers - Systems from the Bureau of Labor Statistics Occupational Employment Statistics (Adjusted to 2009 dollars using the Consumer Price Index) vs. Year

| Year | Computer Programmers | Computer Software Engineers—Applications | Computer Software Engineers—Systems Software |
|---|---|---|---|
| 1999 | $26.42 ($33.73) | $31.62 ($40.37) | $31.84 ($40.65) |
| 2000 | $30.23 ($36.20) | $33.80 ($41.75) | $34.80 ($42.99) |
| 2001 | $30.23 ($36.31) | $34.79 ($41.79) | $35.81 ($43.01) |
| 2002 | $30.62 ($36.20) | $35.48 ($41.95) | $36.46 ($43.11) |
| 2003 | $31.01 ($35.85) | $36.42 ($42.10) | $37.69 ($43.57) |
| 2004 | $35.24 ($39.68) | $39.04 ($43.96) | $40.94 ($46.10) |
| 2005 | $37.22 ($40.54) | $40.09 ($43.66) | $42.22 ($45.98) |
| 2006 | $33.42 ($35.26) | $39.42 ($41.59) | $41.95 ($44.26) |
| 2007 | $34.62 ($35.52) | $41.18 ($42.25) | $43.65 ($44.78) |

Table 1. National Mean Hourly Wages for Software Development Jobs from the Bureau of Labor Statistics (Wages in 2009 dollars are in parentheses)

## B. MAINTENANCE COSTS

Overall, the three systems selected for analysis, the AN/BQQ-10 sonar system, the AN/SQQ-89 surface combat system, and the AN/BYG-1 combat control system exhibit downward trends in the annual maintenance expenditures per unit in service. Costs were analyzed beginning with fiscal year 2000 or the earliest available data from the Navy's VAMOSC database. The following sections explain the results of the analysis for each system. The annual maintenance costs per unit were plotted per fiscal year for the AN/BQQ-10 sonar system, the AN/SQQ-89 surface combat system, and the AN/BYG-1 combat control system, respectively.

The actual cost values and number of units are considered government and business sensitive and have been removed.

### 1. AN/BQQ–10 Sonar System

The AN/BQQ–10 annual software related maintenance costs for fiscal years 2001 to 2007 are illustrated in Figure 5. As expected, the AN/BQQ–10 cost data exhibit a downward trend in the annual software related maintenance costs. Note that the cost trend for the last four years differs significantly from the first three years. The downward slope for fiscal years 2004 to 2007 is not as great and the costs are higher than 2003, but more in line with 2002. The primary reason for this disparity is the absence of any Engineering and Technical Services (ETS) costs for fiscal year 2003. The VAMOSC website includes a section that explains discrepancies and anomalies for data, but no explanation is given. Therefore, either no cost was incurred, or no data were provided. The same issue exists for fiscal year 2002 software support data. This lack of data helps to explain the unusual shape of the plot.

The AN/BQQ–10 annual percentage changes for software related maintenance costs for fiscal years 2001 to 2007 are illustrated in Table 2. The table begins in fiscal year 2002 because no data prior to fiscal year 2001 are available. A negative sign indicates a reduction in costs for the previous fiscal year. With the exception of fiscal year 2004, each year exhibits a decrease in costs. The discrepancy in ETS data for fiscal

year 2003 described above explains the large percentage increase from fiscal year 2003 to fiscal year 2004 as well as the large percentage decrease from fiscal year 2002 to fiscal year 2003. The discrepancy in software support data for fiscal year 2002 also helps to explain the larger than average change from fiscal year 2001 to fiscal year 2002. Taking the average percentage change from fiscal years 2004 to 2007 gives a value of -12.04%, which is more consistent with the overall trend displayed in the graph.

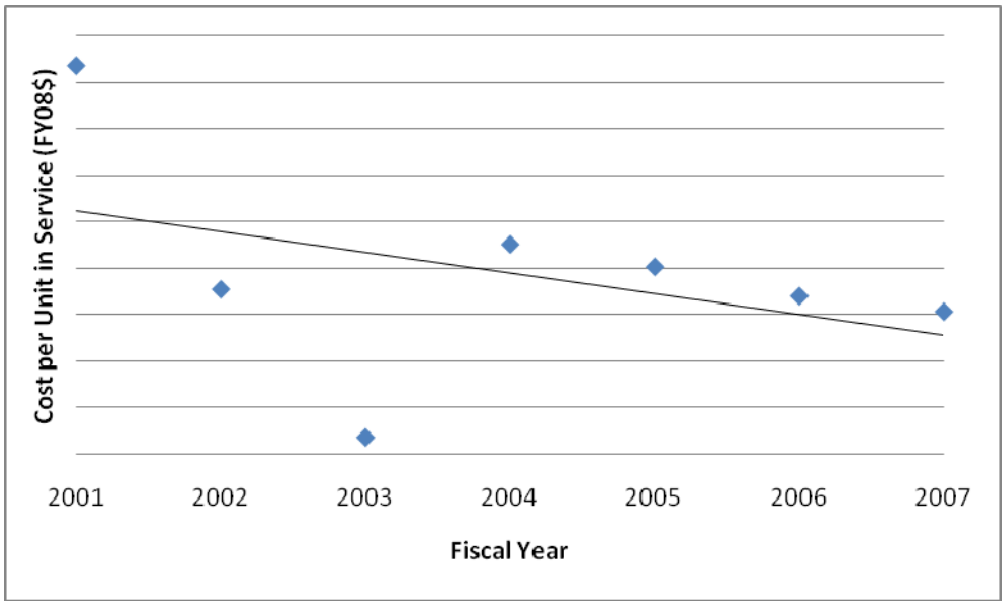


Figure 5. AN/BQQ–10 Software Support and Engineering and Technical Services Cost per Unit (FY08$) vs. Fiscal Year

| Fiscal Year | Annual Percentage Change |
|---|---|
| 2001 | --- |
| 2002 | -57.58% |
| 2003 | -89.77% |
| 2004 | 1144.66% |
| 2005 | -10.69% |
| 2006 | -15.64% |
| 2007 | -9.78% |
| Average | 160.20% |

Table 2. AN/BQQ–10 Software Support and Engineering and Technical Services Cost per Unit (FY08$) Annual Percentage Change

### 2. AN/SQQ–89 Surface Combat System

The AN/SQQ–89 annual software related maintenance costs for fiscal years 2000 to 2007 are illustrated in Figure 6. The data exhibit a well-behaved downward linear trend. The only exception to this trend is fiscal year 2004, where there was a slight increase in costs.

The AN/SQQ–89 annual percentage changes for software related maintenance support costs for fiscal years 2000 to 2007 are illustrated in Table 3. The data in this table also support the trends in lowering costs per system. For the time period analyzed, annual maintenance expenditures per unit decreased by an average of 13.31 percent.

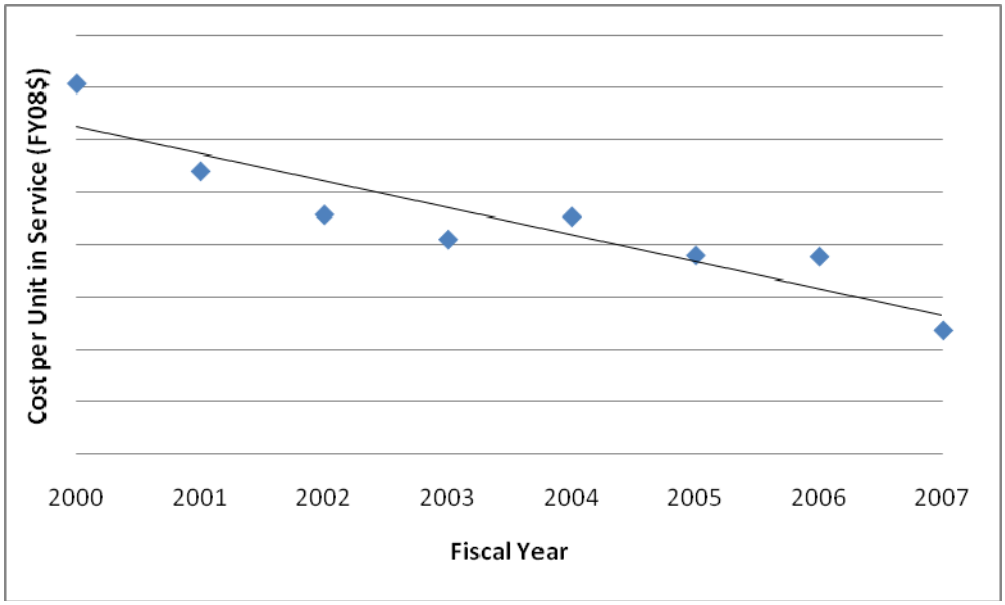


Figure 6. AN/SQQ–89 Software Support and Engineering and Technical Services Cost per Unit (FY08$) vs. Fiscal Year

| Fiscal Year | Annual Percentage Change |
|---|---|
| 2000 | --- |
| 2001 | -23.53% |
| 2002 | -15.46% |
| 2003 | -10.32% |
| 2004 | 10.58% |
| 2005 | -16.31% |
| 2006 | -0.62% |
| 2007 | -37.50% |
| Average | -13.31% |

Table 3. AN/SQQ–89 Software Support and Engineering and Technical Services Cost per Unit (FY08$) Annual Percentage Change

### 3. AN/BYG–1 Combat Control System

The AN/BYG–1 annual software related maintenance costs for fiscal years 2004 to 2007 are illustrated in Figure 7. As expected, the cost data exhibit a downward trend. The large drop in costs from fiscal year 2006 to fiscal year 2007 is explained on the VAMOSC website and is due to a change in how the program office allocated and reported costs. This resulted in a large decrease in the ETS costs for fiscal year 2007 compared to prior years (Naval Center for Cost Analysis [NCCA], 2009a). No retroactive changes to the data were made to reflect the effect that this change would have on previous years. This is described in the FY94—FY08 Ships Anomalies documentation on the VAMOSC site.

The AN/BYG–1 annual percentage changes for software related maintenance support costs for fiscal years 2004 to 2007 are illustrated in Table 4. The data in this table also support the trend in lowering costs per system. One discrepancy in the data is the large decrease from fiscal year 2006 to fiscal year 2007. This change is due to the decrease in ETS costs described in the previous paragraph. For the time period analyzed, annual software related maintenance expenditures per unit decreased by an average of 60.41 percent.

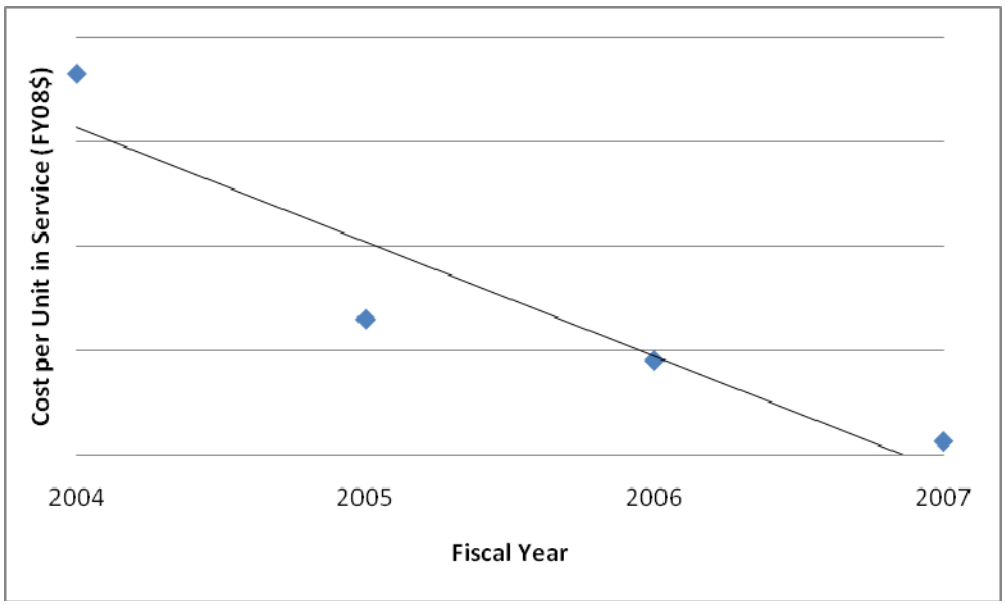


Figure 7. AN/BYG–1 Software Support and Engineering and Technical Services Cost per Unit (FY08$) vs. Fiscal Year

| Fiscal Year | Annual Percentage Change |
|---|---|
| 2004 | --- |
| 2005 | -64.51% |
| 2006 | -30.53% |
| 2007 | -86.19% |
| Average | -60.41% |

Table 4. AN/BYG–1 Software Support and Engineering and Technical Services Cost per Unit (FY08$) Annual Percentage Change

### 4. Summary

The data analyzed for the AN/BQQ–10, AN/SQQ-89, and AN/BYG–1 systems illustrate downward trends as expected. Figure 5, the graph of AN/BQQ–10 software related maintenance costs is difficult to interpret due to the lack of costs associated with ETS in fiscal year 2003 and software support in fiscal year 2002. The reason for the absence of these costs is not explained. However, from fiscal year 2004 to fiscal year 2007 the data behave as expected and illustrate a downward trend. Table 2, the annual percentage change in costs for the AN/BQQ–10 also reflects this trend.

The data for the AN/SQQ–89 software related maintenance costs shown in Figure 6 exhibit a well-behaved downward linear trend. Costs decreased consistently with the exception of fiscal year 2004, where they exhibited a slight increase. The annual percentage changes in costs, shown in Table 3, also illustrate an overall reduction in maintenance costs per system.

Data for the AN/BYG–1 software related maintenance costs shown in Figure 7 also illustrate a clear downward trend. The significant decrease from fiscal year 2006 to fiscal year 2007 is explained by the VAMOSC website as a change in the allocation and reporting of costs by the program office. This resulted in a significant decrease in ETS costs. No data were provided to determine the effect of this change on previous years. The annual percentage change in costs, shown in Table 4, also illustrates an overall reduction in maintenance costs per system.

## C. TRAINING COSTS

The three systems selected for analysis, the AN/BQQ–10 sonar system, the AN/SQQ–89 surface combat system, and the AN/BYG-1 combat control system exhibit varying trends in the annual training costs per unit in service and per individual trained. Costs associated with the AN/BQQ–10 have been increasing, while costs for the AN/SQQ–89 and AN/BYG–1 have been decreasing. Costs were analyzed beginning with fiscal year 2000 or the earliest available data from the Navy's VAMOSC database. The following sections explain the results of the analysis for each system. The annual training

costs per unit and per individual trained were plotted per fiscal year for the AN/BQQ–10 sonar system, the AN/SQQ–89 surface combat system, and the AN/BYG-1 combat control system, respectively.

The actual cost values and number of units are considered government and business sensitive and have been removed.

### 1. AN/BQQ–10 Sonar System

The AN/BQQ–10 training costs per unit for fiscal years 2001 to 2007 are illustrated in Figure 8. The costs initially decreased, but from fiscal years 2002 to 2007 they have exhibited a steady upward linear trend. Although this is contrary to the expected trend, it is indicative of a factor that was not initially anticipated. As more systems are installed on ships, they replace existing systems. Costs are allocated to each system based upon the costs of the learning center and the instruction days and number of graduates associated with a specific system. Therefore, as the number of systems increases, the number of required courses increases, and the fixed costs allocated to that system may also increase. During the time period analyzed, the AN/BQQ–10 has been replacing other existing shipboard sonar systems. Allocating all of the costs of sonar related courses to the AN/BQQ–10 instead of dispersing them among several systems may explain the appearance of rising training costs per unit.

The AN/BQQ–10 training costs per individual trained for fiscal years 2001 to 2007 are illustrated in Figure 9. The costs exhibit an upward linear trend with the exception of fiscal year 2001 and fiscal year 2005. The reason for this disparity appears to be the number of personnel trained. The number of individuals trained for these two years is illustrated in Table 6, and was significantly lower than the other years analyzed. Fiscal year 2006 and fiscal year 2007 indicate that the cost per individual is beginning to stabilize. The overall rise in costs can also be explained by the situation described in the previous paragraph. As the number of systems increases, the allocation of fixed costs to a specific system may also increase.

Table 5 and Table 6 illustrate the annual percentage change for training cost per unit and training cost per individual trained, respectively. The data are consistent with the results displayed on Figures 8 and 9.

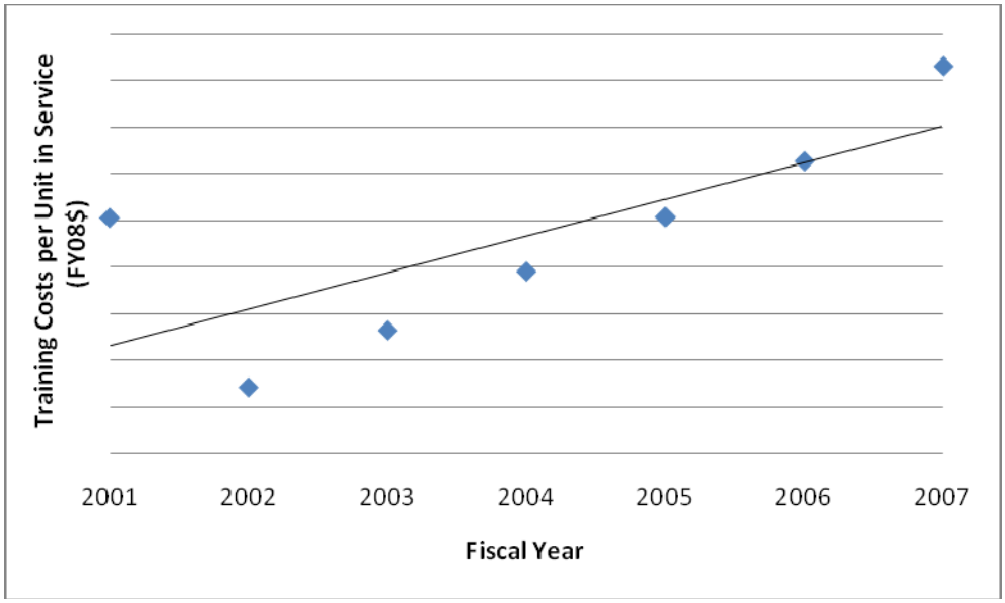


Figure 8. AN/BQQ–10 Training Costs per Unit (FY08$) vs. Fiscal Year

| Fiscal Year | Annual Percentage Change |
|---|---|
| 2001 | ----- |
| 2002 | -72.15% |
| 2003 | 86.96% |
| 2004 | 48.17% |
| 2005 | 30.05% |
| 2006 | 23.72% |
| 2007 | 32.58% |
| Average | 24.89% |

Table 5. AN/BQQ–10 Training Costs per Unit (FY08$) Annual Percentage Change

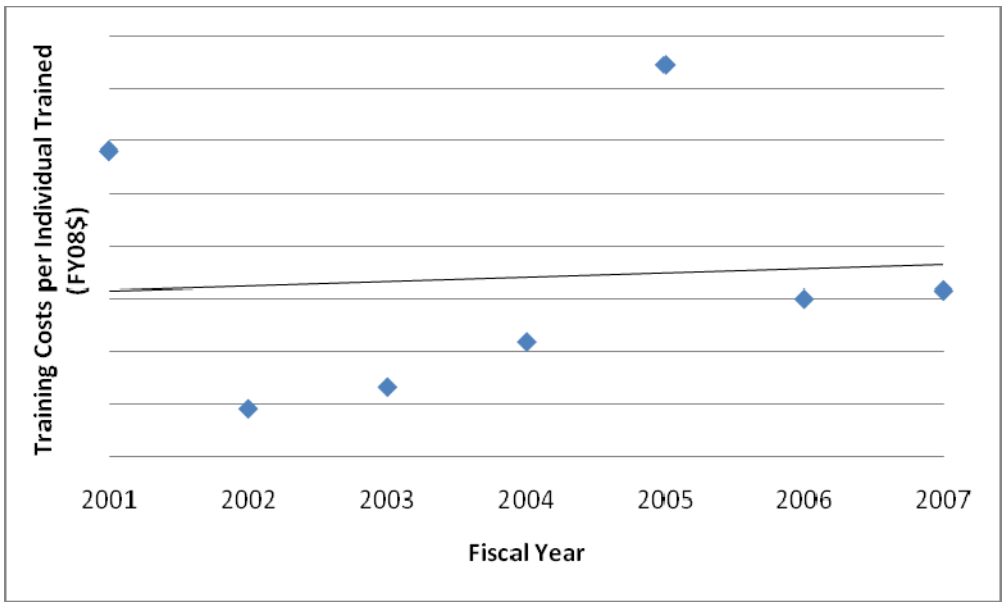


Figure 9. AN/BQQ–10 Training Costs per Individual Trained (FY08$) vs. Fiscal Year

| Fiscal Year | Annual Percentage Change |
|---|---|
| 2001 | ----- |
| 2002 | -84.38% |
| 2003 | 46.34% |
| 2004 | 63.93% |
| 2005 | 241.42% |
| 2006 | -59.59% |
| 2007 | 5.13% |
| Average | 35.48% |

Table 6. AN/BQQ–10 Training Costs per Individual Trained (FY08$) Annual Percentage Change

### 2. AN/SQQ–89 Combat Control System

The AN/SQQ–89 training costs per unit for fiscal years 2000 to 2007 are illustrated in Figure 10. Overall, the costs exhibit a downward trend. However, there is significant variation in the annual costs. Table 7, the annual percentage change in the costs illustrates this variation. Although the average for fiscal years 2000 to 2007 is a 0.37 percent decrease, annual changes range from a 50.17 percent increase to a 34.21 percent decrease.

The AN/SQQ–89 training costs per individual trained for fiscal years 2000 to 2007, illustrated in Figure 11, exhibit a much more defined trend. The costs decrease exponentially and appear to have steadied. The annual percentage changes, illustrated in Table 8, further show this downward trend. Overall costs decreased by an average of 12.14 percent. Like the AN/BQQ–10, the highest costs per individual trained occurred in years where the fewest personnel were trained.

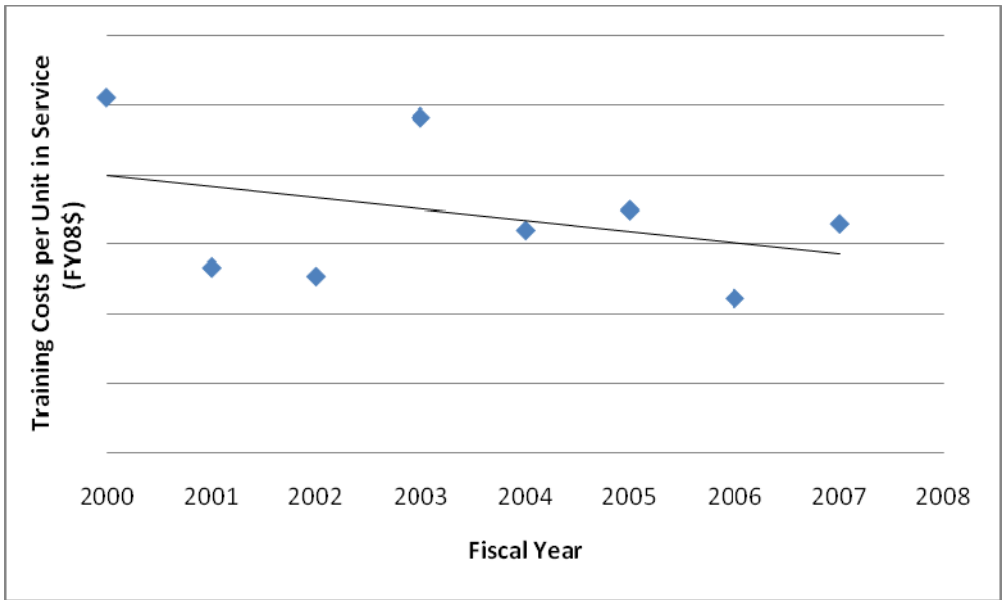


Figure 10. AN/SQQ–89 Training Costs per Unit (FY08$) vs. Fiscal Year

| Fiscal Year | Annual Percentage Change |
|---|---|
| 2000 | ----- |
| 2001 | -34.21% |
| 2002 | -2.70% |
| 2003 | 50.10% |
| 2004 | -23.71% |
| 2005 | 5.54% |
| 2006 | -23.07% |
| 2007 | 25.45% |
| Average | -0.37% |

Table 7. AN/SQQ–89 Training Costs per Unit (FY08$) Annual Percentage Change

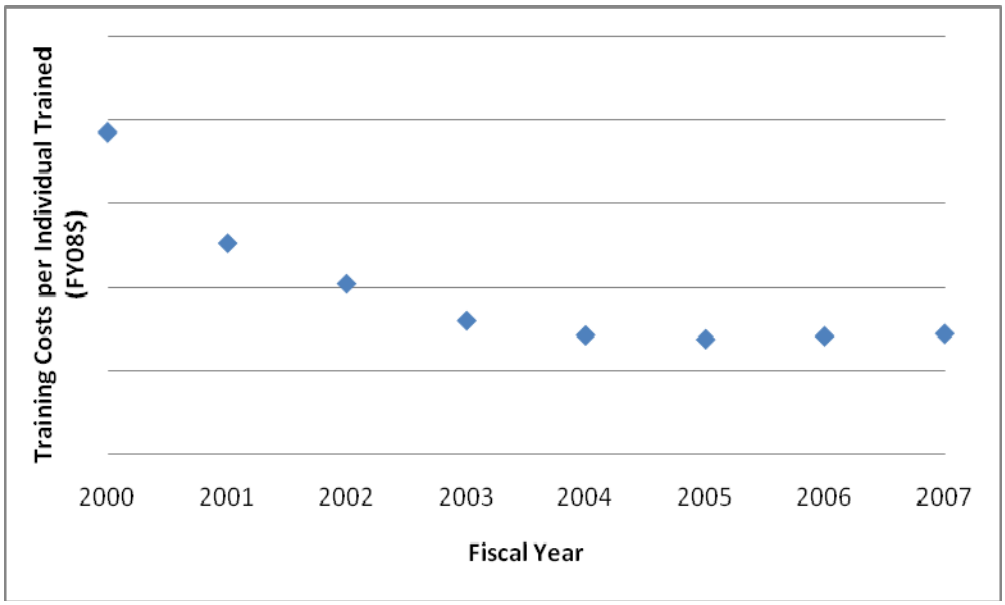


Figure 11. AN/SQQ–89 Training Costs per Individual Trained (FY08$) vs. Fiscal Year

| Fiscal Year | Annual Percentage Change |
|---|---|
| 2000 | ----- |
| 2001 | -34.58% |
| 2002 | -19.19% |
| 2003 | -21.66% |
| 2004 | -10.96% |
| 2005 | -3.30% |
| 2006 | 2.63% |
| 2007 | 2.08% |
| Average | -12.14% |

Table 8. AN/SQQ–89 Training Costs per Individual Trained (FY08$) Annual Percentage Change

### 3. AN/BYG–1 Combat Control System

The AN/BYG–1 training costs per unit for fiscal years 2004 to 2007 are illustrated in Figure 12. Overall, the shape of the curve exhibits an exponential decrease with costs steadying during the last three years. Table 9, the annual percentage change in training costs per unit illustrates this downward trend. On average, costs have decreased by 36.94 percent.

AN/BYG–1 training costs per individual trained for fiscal years 2004 to 2007, illustrated in Figure 13, initially increased significantly, but have exhibited a well-behaved downward linear trend from fiscal years 2005 to 2007. Again, the cost per individual trained appears to decrease as the number of personnel trained increases. Table 10, annual percentage change in costs per individual trained, supports this conclusion. Although the average of the data analyzed illustrates an increase of 11.94 percent, this number is skewed by the 90.82 percent increase from fiscal year 2004 to fiscal year 2005.

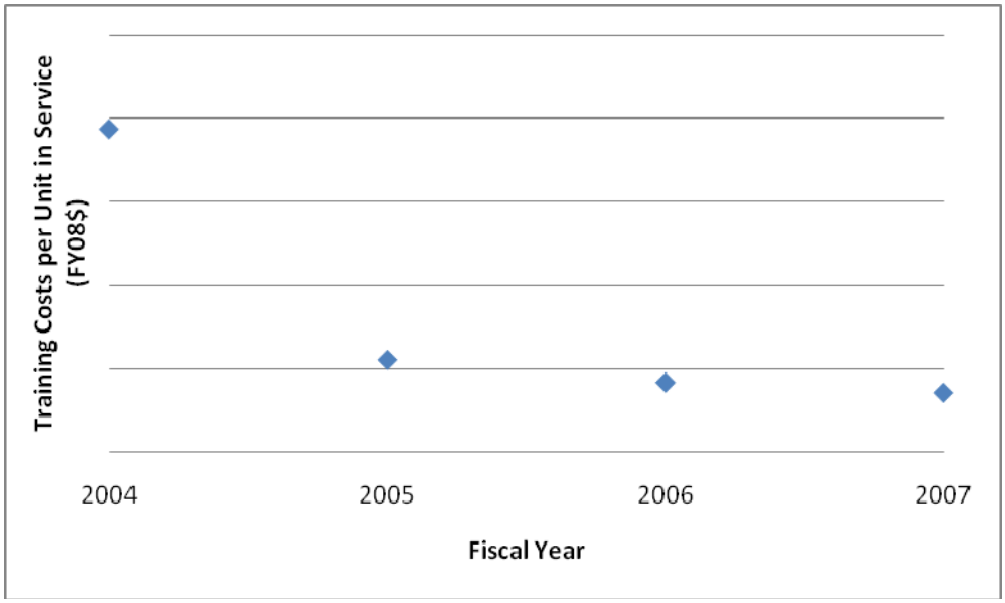


Figure 12. AN/BYG–1 Training Costs per Unit (FY08$) vs. Fiscal Year

| Fiscal Year | Annual Percentage Change |
|---|---|
| 2004 | ----- |
| 2005 | -71.59% |
| 2006 | -24.44% |
| 2007 | -14.78% |
| Average | -36.94% |

Table 9. AN/BYG–1 Training Costs per Unit (FY08$) Annual Percentage Change

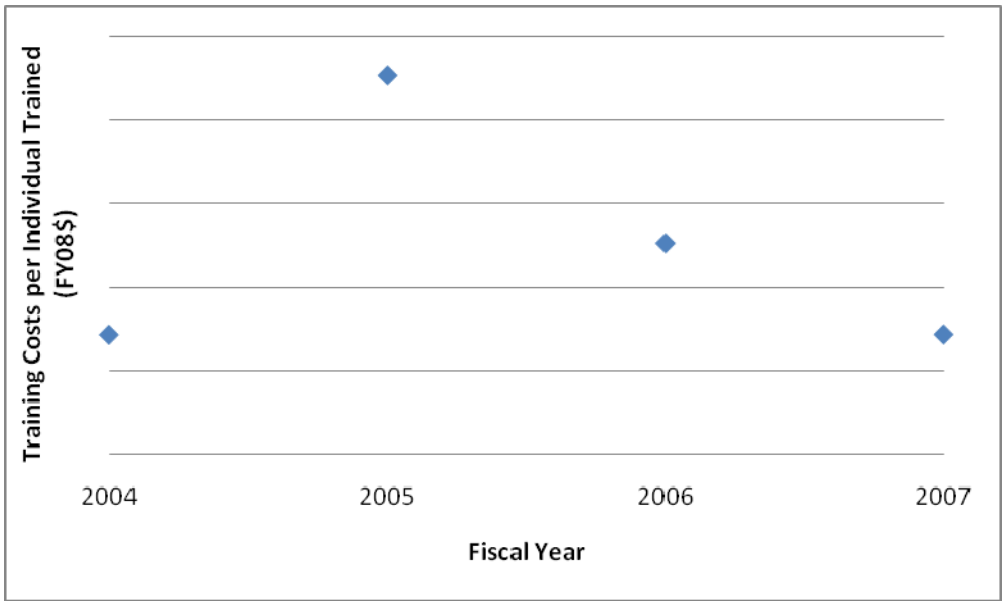


Figure 13. AN/BYG–1 Training Costs per Individual Trained (FY08$) vs. Fiscal Year

| Fiscal Year | Annual Percentage Change |
|---|---|
| 2004 | ----- |
| 2005 | 90.82% |
| 2006 | -30.78% |
| 2007 | -24.21% |
| Average | 11.94% |

Table 10. AN/BYG–1 Training Costs per Individual Trained (FY08$) Annual Percentage Change

### 4. Summary

The three systems analyzed illustrated varying trends in both the training costs per unit and training costs per individual trained. The AN/BQQ–10 exhibited a well-behaved upward linear trend in training costs per unit and a less defined upward trend in training costs per individual trained. The AN/SQQ–89 exhibited no significant trends in training costs per unit, but exhibited an exponential decrease in training costs per individual trained. The AN/BYG–1 exhibited an exponential decrease in costs per unit and a well-behaved downward linear trend from fiscal years 2005 to 2007 for training costs per individual trained.

A common factor in the costs for each system is that costs per individual trained decrease as more personnel are trained. This suggests that the fixed costs of operating the training courses are the primary cost drivers. As output increases, the cost per unit of output is lower. The results of this analysis also seem to indicate that costs can only be decreased to a certain level. Therefore, greater commonality of systems will be successful in reducing training costs to the point where it can reduce the fixed costs of operating courses.

Another important note is the trend illustrated by Figure 8, the AN/BQQ–10 training costs per unit. As the number of units increased, the costs per unit increased. This may be due to the allocation of fixed costs associated with operating the training centers. Therefore, using cost per unit may not be a good indication of the effect of commonality on training costs until all systems have been converted. A better analysis would concentrate on the total cost required to train sonar operators across the fleet.

## D. RESEARCH, DEVELOPMENT, TEST, AND EVALUATION COSTS

Overall, the costs for the following RDT&E budget items exhibit varying trends in funding.

- Budget Activity (BA)– 4, Program Element 0603561N/Advanced Submarine System Development—Project Unit 0223/Sub Combat System Improvement (ADV)

- Budget Activity (BA)—4, Program Element 0603553N/Surface ASW—Project Unit 1704/Undersea Warfare
- Budget Activity (BA)—5, Program Element 0604503N/Submarine System Equipment Development—Project Unit 0219/Sub Sonar Improvement (ENG)
- Budget Activity (BA)—5, Program Element 0604562N/Submarine Tactical Warfare System—Project Unit 0236/SSN Comb Cont Sys Imprvmnt (ENG)
- Budget Activity (BA)—5, Program Element 0604261N/Acoustic Search Sensors—Project Unit 0480/ASW Sensors & Processors
- Budget Activity (BA)—7, Program Element 0205620N/Surface ASW Combat System Integration—Project Unit 0896/AN/SQQ-89 Modification and Budget Activity (BA)—7, Program Element 0205620N/Surface ASW Combat System Integration—Project Unit 1916/Surface ASW System Improvement

RDT&E costs were analyzed due to the belief that this is the phase of production where software reuse would have the most significant effect. This belief was developed through interviews and a literature review. Empirical data relating to specific cost avoidance from software reuse within each program were not available. General percentages of reuse from the APBs developed under Program Element 0603561N/ Advanced Submarine System Development were estimated by PEO IWS 5, but do not necessarily reflect actual reuse percentages. Therefore the data used for analysis do not take into account specific instances of software reuse. Increases and decreases in funding from year to year may have been affected by other factors not related to software reuse. Also, as discussed in Chapter II, software reuse may not result in savings. Depending on the information related to the assets available and how they are reused, software reuse has the potential to increase costs.

The analysis conducted simply uses available cost data and anecdotal evidence that software reuse does occur at some level within the programs chosen and identifies trends in the funding. Actual effects of software reuse may be different than the trends exhibited by the data. The following sections explain the results of the analysis for each program element.

### 1. Trend Analysis of RDT&E Costs

#### *a. Program Element 0603561N/Advanced Submarine System Development*

Trends in the RDT&E costs for Program Element 0603561N/Advanced Submarine System Development—Project Unit 0223/Sub Combat System Improvement (ADV) for FY 2000 to 2008 are illustrated in Figure 14. As expected they exhibit a downward trend. Two notable exceptions are fiscal years 2004 and 2005. Funding for the program experienced a significant decrease in fiscal year 2004 and a subsequent increase in fiscal year 2005. The fiscal year 2004 decrease was due to a reprogramming of $32,655,000 and illustrates one of the other factors that can affect program funding (DoN FM&C, 2003).

Results of the annual percentage change in funding are displayed in Table 11. The table illustrates large variations in the year-to-year funding. There are examples of large increases and large decreases from year to year. Although the trend in costs appears to be in the downward direction, the average annual percentage change in annual funding is a 0.68 percent increase.

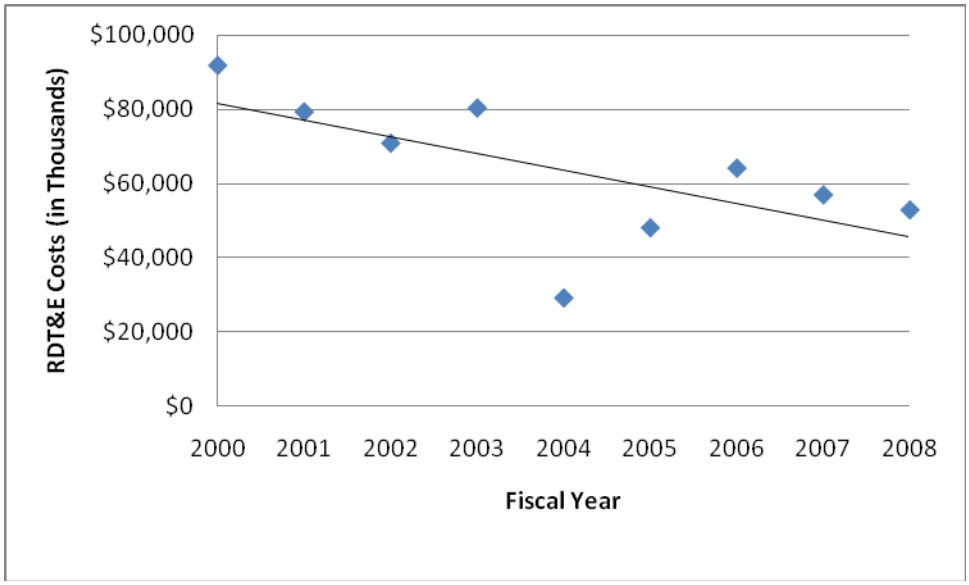


Figure 14. Program Element 0603561N/Advanced Submarine System Development—Project Unit 0223/Sub Combat System Improvement (ADV) RDT&E Costs (FY09$) per Fiscal Year

| Fiscal Year | RDT&E Cost (FY09$) | Annual Percentage Change |
|---|---|---|
| 2000 | $91,661,440 | ------ |
| 2001 | $79,175,646 | -13.62% |
| 2002 | $70,729,533 | -10.67% |
| 2003 | $80,191,460 | 13.38% |
| 2004 | $29,098,167 | -63.71% |
| 2005 | $47,967,494 | 64.85% |
| 2006 | $64,014,113 | 33.45% |
| 2007 | $56,843,747 | -11.20% |
| 2008 | $52,767,820 | -7.17% |
| Average | $91,661,440 | 0.66% |

Table 11. Program Element 0603561N/Advanced Submarine System Development—Project Unit 0223/Sub Combat System Improvement (ADV) RDT&E Costs (FY09$) and Annual Percentage Change

### *b. Program Element 0603553N/Surface ASW*

Trends in the RDT&E costs for Program Element 0603553N/Surface ASW—Project Unit 1704/Undersea Warfare for fiscal years 2000 to 2008 are illustrated in Figure 15. Overall, the costs analyzed exhibit an upward trend. This is not the anticipated result. However, the large increases in funding beginning in fiscal year 2004 can be traced to the incorporation of at-sea testing of the products developed. Further investigation revealed the results of Table 12, which exhibit the annual costs of incorporating at-sea testing into the program. There are also three distinct portions of the graph. From fiscal years 2000 to 2003, the costs exhibit a downward trend. From fiscal years 2004 to 2006, they exhibit a linear increase and from fiscal years 2006 to 2008, they appear to again be decreasing.

Results of the annual percentage change in funding are displayed in Table 13. The table illustrates that there have been large variations in the annual funding for this program. The average of the annual percentage changes is 36.50 percent, which is a significant increase from year to year.

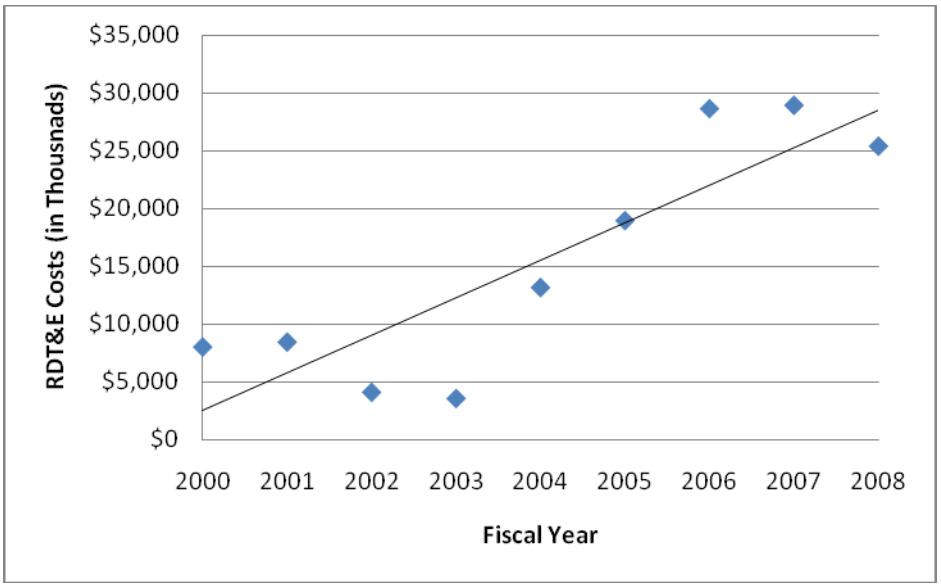


Figure 15. Program Element 0603553N/Surface ASW—Project Unit 1704/Undersea Warfare RDT&E Costs (FY09$) per Fiscal Year

| Fiscal Year | Cost of At-Sea Testing (FY09$) | Percent of Total Costs |
|---|---|---|
| 2004 | $10,307,274 | 78.23% |
| 2005 | $16,265,390 | 85.81% |
| 2006 | $26,391,465 | 92.19% |
| 2007 | $16,141,882 | 55.83% |
| 2008 | $20,656,265 | 81.36% |

Table 12. Annual Costs of At-Sea Testing (FY09$) for Program Element 0603553N/Surface ASW—Project Unit 1704/Undersea Warfare and Percent of Total Costs per Fiscal Year

| Fiscal Year | RDT&E Cost (FY09$) | Annual Percentage Change |
|---|---|---|
| 2000 | $8,040,708 | ------ |
| 2001 | $8,462,084 | 5.24% |
| 2002 | $4,128,201 | -51.22% |
| 2003 | $3,586,632 | -13.12% |
| 2004 | $13,176,251 | 267.37% |
| 2005 | $18,956,113 | 43.87% |
| 2006 | $28,628,735 | 51.03% |
| 2007 | $28,914,029 | 1.00% |
| 2008 | $25,388,195 | -12.19% |
| Average | $15,475,661 | 36.50% |

Table 13. Program Element 0603553N/Surface ASW—Project Unit 1704/Undersea Warfare RDT&E Costs (FY09$) and Annual Percentage Change

### *c. Program Element 0604503N/Submarine System Equipment Development*

Trends in the RDT&E costs for Program Element 0604503N/Submarine System Equipment Development—Project Unit 0219/Sub Sonar Improvement (ENG) for fiscal years 2000 to 2008 are illustrated in Figure 16. As expected, the costs exhibit an overall downward trend. Fiscal year 2002 costs also include Project Unit 9070/MPP/SPB/A-RCI Model for Tactical Control Info Mgmt. This project number has the same R-2a description and justification, but is listed separately. No explanation is provided for the additional entry and the item is consolidated into Project Unit 0219 in fiscal year 2003.

Results of calculating the annual percentage change are displayed in Table 14. The table illustrates that there have been large variations in annual funding, but overall there has been an average decrease of 2.04 percent.

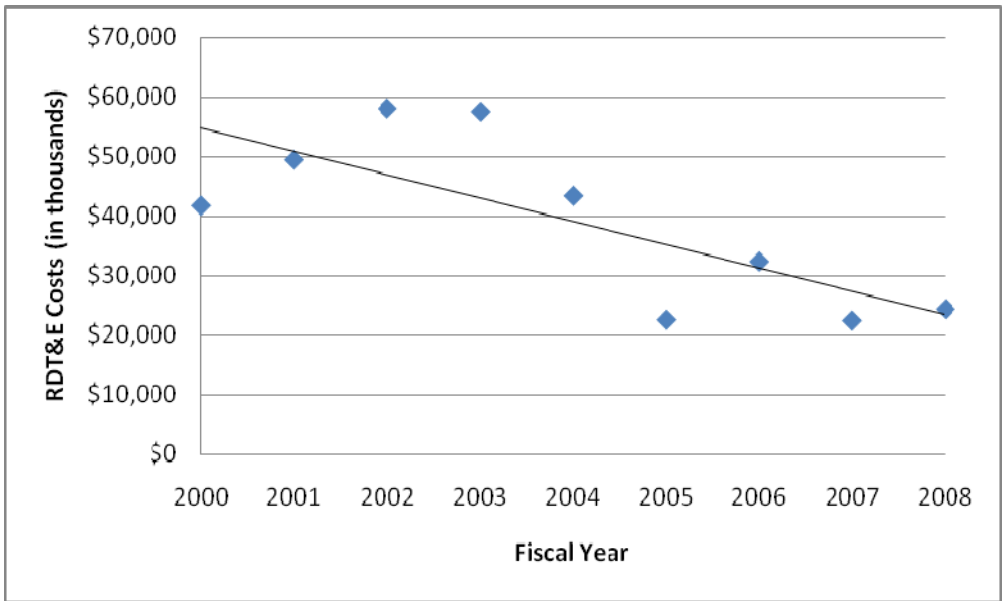


Figure 16. Program Element 0604503N/Submarine System Equipment Development—Project Unit 0219/Sub Sonar Improvement (ENG) RDT&E Costs (FY09$) per Fiscal Year

| Fiscal Year | RDT&E Cost (FY09$) | Annual Percentage Change |
|---|---|---|
| 2000 | $41,749,968 | ------ |
| 2001 | $49,397,990 | 18.32% |
| 2002 | $57,968,474 | 17.35% |
| 2003 | $57,423,040 | -0.94% |
| 2004 | $43,385,355 | -24.45% |
| 2005 | $22,631,783 | -47.84% |
| 2006 | $32,389,186 | 43.11% |
| 2007 | $22,497,813 | -30.54% |
| 2008 | $24,439,170 | 8.63% |
| Average | $39,098,086 | -2.04% |

Table 14. Program Element 0604503N/Submarine System Equipment Development—Project Unit 0219/Sub Sonar Improvement (ENG) RDT&E Costs (FY09$) and Annual Percentage Change

### *d. Program Element 0604562N/Submarine Tactical Warfare System—Project Unit 0236/SSN Comb Cont Sys Imprvmnt (ENG)*

Trends in the RDT&E costs for Program Element 0604503N/Submarine System Equipment Development—Project Unit 0219/Sub Sonar Improvement (ENG) for fiscal years 2000 to 2008 are illustrated in Figure 17. For fiscal year 2002, there was no breakdown of costs available in Exhibit R-3, so 65 percent of the total cost of the program was used. This corresponds to the average percentage of software related costs for the program between fiscal years 2004 and 2008. Although the overall trend is upward, it is difficult to draw any conclusions due to the spread of the data. From fiscal years 2001 to 2006 there is a slight downward linear trend. In fiscal year 2007, the costs increase by 48.46 percent and then begin to slope downward again. Overall it seems that the upward trend is driven by the increases from fiscal years 2000 to 2001 and fiscal years 2006 to 2007.

Results of calculating the annual percentage change, displayed in Table 15, also illustrate this inconsistency. There are large annual variations and the overall average indicates a 36.62 percent increase. However, removing the largest increase, 258.66 percent from fiscal years 2000 to 2001 lowers the average to 4.90 percent.

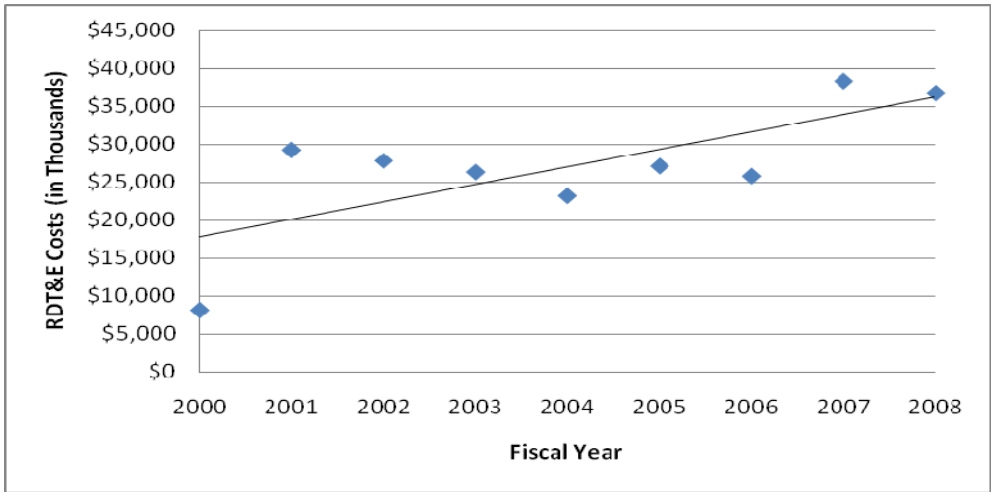


Figure 17. Program Element 0604562N/Submarine Tactical Warfare System—Project Unit 0236/SSN Comb Cont Sys Imprvmnt (ENG) RDT&E Costs (FY09$) per Fiscal Year

| Fiscal Year | RDT&E Cost (FY09$) | Annual Percentage Change |
|---|---|---|
| 2000 | $8,167,484 | ------ |
| 2001 | $29,293,638 | 258.66% |
| 2002 | $27,876,487 | -4.84% |
| 2003 | $26,383,902 | -5.35% |
| 2004 | $23,255,003 | -11.86% |
| 2005 | $27,187,853 | 16.91% |
| 2006 | $25,855,630 | -4.90% |
| 2007 | $38,385,042 | 48.46% |
| 2008 | $36,805,930 | -4.11% |
| Average | $27,023,441 | 36.62% |

Table 15. Program Element 0604562N/Submarine Tactical Warfare System—Project Unit 0236/SSN Comb Cont Sys Imprvmnt (ENG) RDT&E Costs (FY09$) and Annual Percentage Change

#### *e. Program Element 0604261N/Acoustic Search Sensors—Project Unit 0480/ASW Sensors & Processors*

Trends in the RDT&E costs for Program Element 0604261N/Acoustic Search Sensors—Project Unit 0480/ASW Sensors & Processors for fiscal years 2000 to 2008 are illustrated in Figure 18. As expected, the data exhibit a downward linear trend. The trend is consistent with the exception of fiscal years 2006 to 2008. The large increase in fiscal year 2008 is due to starting a new program. Exhibit R-2a explains that the purpose of the program is to "develop a coherent source that will satisfy the search and localization requirement in the harsh, shallow water littorals" (DoN FM&C, 2007).

Results of calculating the annual percentage change are displayed in Table 16. Although there is large variation in the annual percentages, the average is a 2.33 percent decrease.

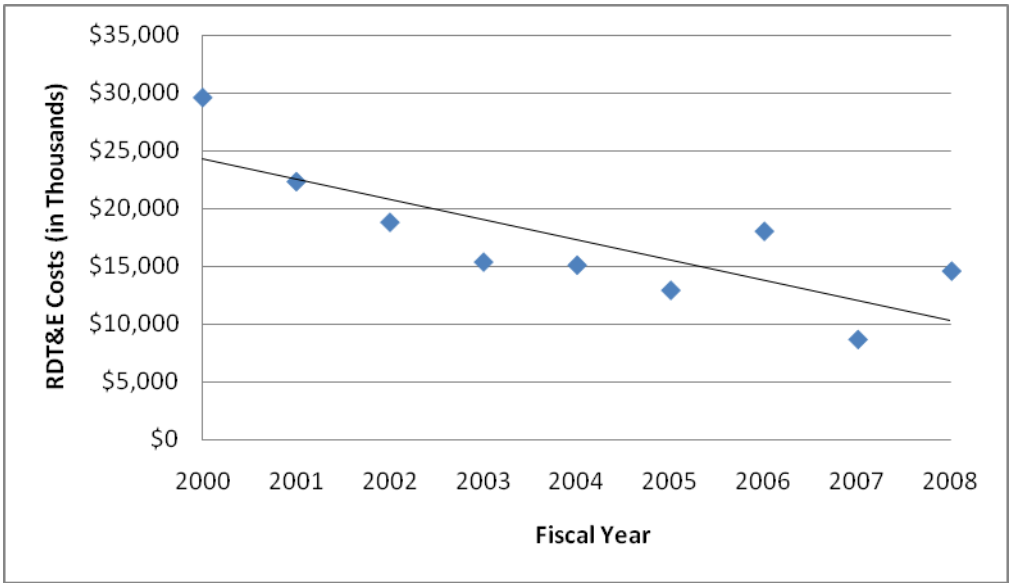


Figure 18. Program Element 0604261N/Acoustic Search Sensors—Project Unit 0480/ASW Sensors & Processors RDT&E Costs (FY09$) per Fiscal Year

| Fiscal Year | RDT&E Cost (FY09$) | Annual Percentage Change |
|---|---|---|
| 2000 | $29,639,272 | ------ |
| 2001 | $22,347,046 | -24.60% |
| 2002 | $18,811,170 | -15.82% |
| 2003 | $15,360,894 | -18.34% |
| 2004 | $15,094,936 | -1.73% |
| 2005 | $12,909,966 | -14.47% |
| 2006 | $18,033,723 | 39.69% |
| 2007 | $8,639,493 | -52.09% |
| 2008 | $14,578,445 | 68.74% |
| Average | $17,268,327 | -2.33% |

Table 16. Program Element 0604261N/Acoustic Search Sensors—Project Unit 0480/ASW Sensors & Processors RDT&E Costs (FY09$) and Annual Percentage Change

### *f. Program Element 0205620N/Surface ASW Combat System Integration and Program Element 0205620N/Surface ASW System Improvement*

Trends in the RDT&E costs for Program Element 0205620N/Surface ASW Combat System Integration—Project Unit 0896/AN/SQQ-89 Modification and Program Element 0205620N/Surface ASW Combat System Integration—Project Unit 1916/Surface ASW System Improvement for fiscal years 2000 to 2008 are illustrated in Figure 19. As expected, the data exhibit a downward linear trend. In this case, the data illustrate a well-behaved linear trend with the exception of fiscal years 2000 and 2003. The large increase in fiscal year 2003 is attributed to a congressional add of $11.6 million (DoN FM&C, 2003). This again illustrates how funding can be influenced by factors other than process improvements.

Results of calculating the annual percentage change are displayed in Table 17. These results again illustrate a wide range of annual funding changes. Overall, the average is a 0.48 percent increase. Removing the 87.02 percent increase from fiscal year 2000 to fiscal year 2001 drops the average to an 11.89 percent decrease, which is more consistent with the trend identified in Figure 13.

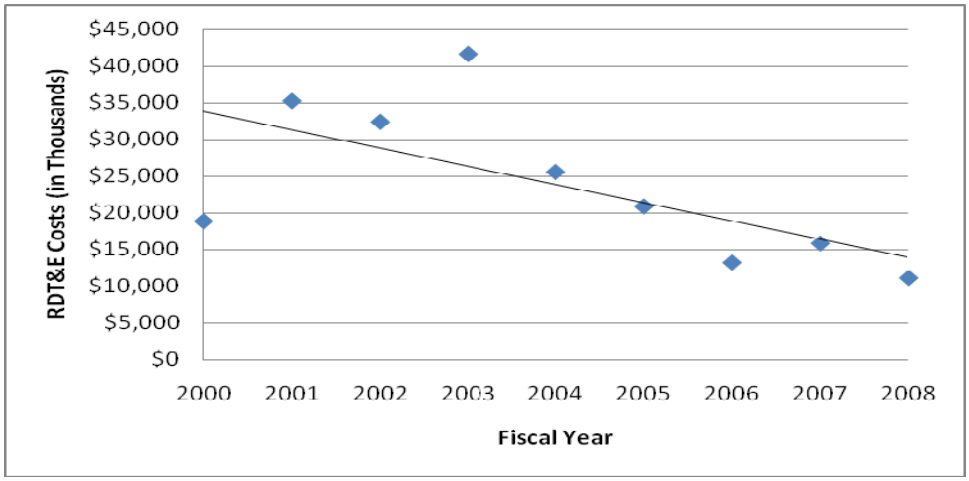


Figure 19. Program Element 0205620N/Surface ASW Combat System Integration—Project Unit 0896/AN/SQQ–89 Modification and Budget Activity (BA)—7 and Program Element 0205620N/Surface ASW Combat System Integration—Project Unit 1916/Surface ASW System Improvement RDT&E Costs (FY09$) per Fiscal Year

| Fiscal Year | RDT&E Cost (FY09$) | Annual Percentage Change |
|---|---|---|
| 2000 | $18,850,156 | ------ |
| 2001 | $35,253,398 | 87.02% |
| 2002 | $32,388,080 | -8.13% |
| 2003 | $41,619,010 | 28.50% |
| 2004 | $25,566,249 | -38.57% |
| 2005 | $20,869,882 | -18.37% |
| 2006 | $13,221,884 | -36.65% |
| 2007 | $15,824,865 | 19.69% |
| 2008 | $11,127,445 | -29.68% |
| Average | $23,857,885 | 0.48% |

Table 17. Program Element 0205620N/Surface ASW Combat System Integration—Project Unit 0896/AN/SQQ–89 Modification and Budget Activity (BA)—7 and Program Element 0205620N/Surface ASW Combat System Integration—Project Unit 1916/Surface ASW System Improvement RDT&E Costs (FY09$) and Annual Percentage Change

### *g. Total Software Related Costs for the Selected Programs*

Trends in the total RDT&E costs for the above programs are illustrated in Figure 20. As expected, the costs exhibit an overall downward trend.

Results of calculating the annual percentage change are displayed in Table 18. These results illustrate a wide range of annual funding changes. Overall, the average is a 0.96 percent decrease. This is consistent with the trend identified.

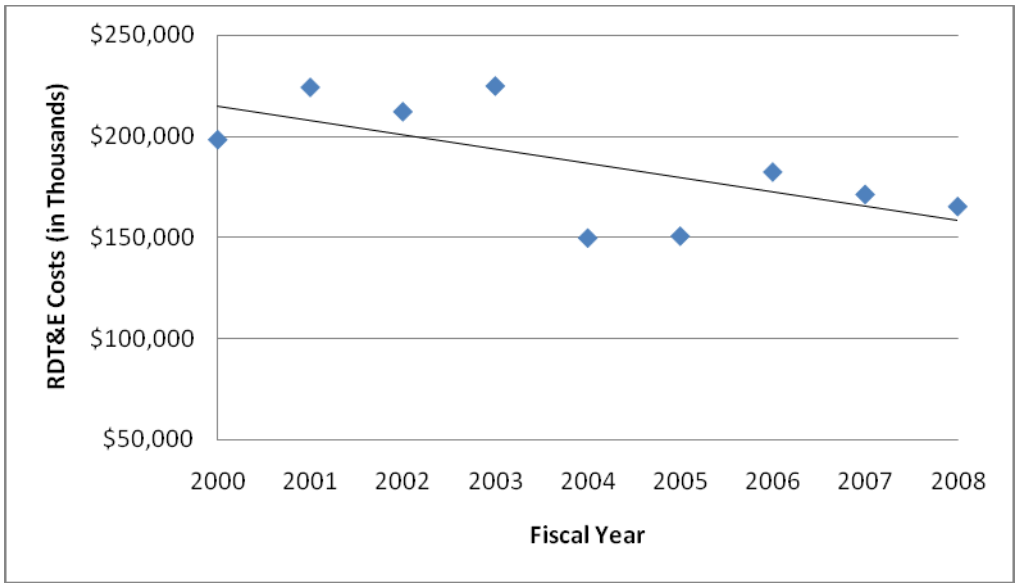


Figure 20. Total Software Related RDT&E Costs (FY09$) per Fiscal Year

| Fiscal Year | RDT&E Cost (FY09$) | Annual Percentage Change |
|---|---|---|
| 2000 | $198,109,028 | ------ |
| 2001 | $223,929,803 | 13.03% |
| 2002 | $211,901,944 | -5.37% |
| 2003 | $224,564,938 | 5.98% |
| 2004 | $149,575,961 | -33.39% |
| 2005 | $150,523,089 | 0.63% |
| 2006 | $182,143,271 | 21.01% |
| 2007 | $171,104,989 | -6.06% |
| 2008 | $165,107,005 | -3.51% |
| Average | $186,328,892 | -0.96% |

Table 18. Total Software Related RDT&E Costs (FY09$) per Fiscal Year

### 2. Total ASW Sonar and Fire Control Related RDT&E Costs

Table 19 illustrates the total fiscal years 2008 and 2009 funding for ASW sonar and fire control related RDT&E. The costs displayed do not differentiate between software and hardware costs. Total RDT&E costs for sonar and fire control related systems in the Department of the Navy budget for fiscal years 2008 and 2009 are $597,457,420 and $624,577,000, respectively. Although the fiscal year 2009 figure

accounts for only 3.23 percent of the total $19,337,238,000 RDT&E funding available for obligation until September 30, 2010, it is still a significant amount.

| Budget Activity | Program Element | Project Title | Fiscal Year 2008 Costs | Fiscal Year 2009 Costs |
|---|---|---|---|---|
| 2 | 0602747N | Undersea Warfare Applied Research | $72,840,460 | $58,658,000 |
| 3 | 0603747N | Undersea Warfare Advanced Technology | $76,553,330 | $81,490,000 |
| 4 | 0603561N | Sub Combat System Improvement (ADV) | $52,767,820 | $47,135,000 |
| 4 | 0603561N | Undersea Superiority | $0 | $36,933,000 |
| 4 | 0603553N | Surface ASW | $47,172,125 | $29,574,000 |
| 4 | 0603581N | LCS Mission Package Development (ASW) | $21,010,500 | $9,731,000 |
| 5 | 0603254N | ADV ASW Sensors & Proc | $3,265,255 | $10,320,000 |
| 5 | 0603506N | Surface Ship Torpedo Defense (SSTD) | $27,862,765 | $49,171,000 |
| 5 | 0604221N | P-3 Sensor Integration | $3,124,170 | $1,460,000 |
| 5 | 0604518N | USW Decision Support | $17,027,640 | $14,792,000 |
| 5 | 0604558N | New Design SSN Combat Sys Dev | $35,119,000 | $28,820,000 |
| 5 | 0604558N | Submarine Multi-Mission Team Trainer | $6,385,365 | $2,786,000 |
| 5 | 0604261N | ASW Sensors & Processors | $18,937,870 | $18,325,000 |
| 5 | 0604503N | Sub Sonar Improvement (ENG) | $61,682,565 | $67,894,000 |
| 5 | 0604562N | SSN Comb Cont Sys Imprvmnt (ENG) | $56,563,920 | $58,592,000 |
| 5 | 0604610N | Lightweight Hybrid Torpedo | $26,899,530 | $50,732,000 |
| 7 | 0205620N | Surface ASW System Improvement | $18,388,755 | $21,720,000 |
| 7 | 0204311N | IUSS Detect/Classify System | $31,605,070 | $20,565,000 |
| 7 | 0205632N | MK 48 ADCAP | $20,251,280 | $15,879,000 |
| | | Total | $597,457,420 | $624,577,000 |

Table 19. Total ASW Sonar and Fire Control Related RDT&E Costs (FY09$)

### 3. Summary

Overall, RDT&E software costs related to ASW sonar and fire control systems exhibit a downward trend, even as the cost of labor, the primary cost driver for software development, has increased. However, only two of the programs analyzed illustrated a well-behaved downward linear pattern. Two of the other programs showed costs generally moving downward, but extremes in each direction make it difficult to come to a definitive conclusion. The final two programs analyzed actually showed costs moving in the upward direction. These programs were also not indicative of a well-defined linear trend.

The overall downward trend does indicate the potential of software reuse cost savings. However, the actual amount of these reductions that can be attributed to software reuse is inconclusive. Specific empirical data regarding reuse within each of the programs were not available, so any conclusion concerning direct cost avoidance from software reuse is speculation.

In addition, the analysis highlighted other potential events that can significantly affect the funding for each program identified. These included congressional additions and reprogrammings. These two examples are also not all-inclusive, but illustrate some of the outside effects that can influence a program's funding. This is important to note when analyzing programs that rely on incremental development. Events independent of the actions within the program can significantly affect annual funding. An over reliance on consistent annual funding to provide basic system capabilities could be problematic.

Calculating the total annual RDT&E costs for sonar- and fire-control related systems for fiscal years 2008 and 2009 illustrates how much money is potentially available within an Anti-Submarine Warfare (ASW) Community of Interest (COI). Making a judgment as to whether this is sufficient funding to validate the existence of such a COI is not the intent of this research. However, there does seem to be a sufficient number of efforts underway to justify future investment to facilitate collaboration.

# V. SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

## A. SUMMARY OF RESULTS

This research analyzed the costs associated with development and support of anti-submarine warfare software to determine the cost-effectiveness of software reuse. The following six program elements were selected from the Department of the Navy's Research, Development, Test and Evaluation (RDT&E) budget in order to analyze trends in software development costs.

- Budget Activity (BA)– 4, Program Element 0603561N/Advanced Submarine System Development—Project Unit 0223/Sub Combat System Improvement (ADV)
- Budget Activity (BA)—4, Program Element 0603553N/Surface ASW—Project Unit 1704/Undersea Warfare
- Budget Activity (BA)—5, Program Element 0604503N/Submarine System Equipment Development—Project Unit 0219/Sub Sonar Improvement (ENG)
- Budget Activity (BA)—5, Program Element 0604562N/Submarine Tactical Warfare System—Project Unit 0236/SSN Comb Cont Sys Imprvmnt (ENG)
- Budget Activity (BA)—5, Program Element 0604261N/Acoustic Search Sensors—Project Unit 0480/ASW Sensors & Processors
- Budget Activity (BA)—7, Program Element 0205620N/Surface ASW Combat System Integration—Project Unit 0896/AN/SQQ-89 Modification and Budget Activity (BA)—7, Program Element 0205620N/Surface ASW Combat System Integration—Project Unit 1916/Surface ASW System Improvement

Four of the six programs exhibited downward trends in annual funding and two exhibited an upward trend. Within the four programs, only two exhibited a well-behaved downward linear trend in costs.

Software related support costs were analyzed for three specific anti-submarine warfare systems, AN/BQQ-10 sonar system, the AN/SQQ-89 surface combat system and the AN/BYG-1 combat control system. The two types of costs selected for analysis were

software related maintenance costs and training costs. Within those two categories, costs were further divided into maintenance costs per unit in service, training costs per unit in service, and training costs per individual trained.

Results from the analysis of software related maintenance costs exhibited a clear downward trend for all three systems. In each case it appears that the cost associated with maintaining systems is decreasing on a per unit basis.

The results obtained from the analysis of training costs are less consistent. The AN/BQQ–10 exhibited upward trends in both training costs per unit and per individual trained. The AN/SQQ–89 and AN/BYG–1 exhibited downward trends for both costs per unit and costs per individual trained. The shape of the graphs was not consistent in either case. The AN/SQQ–89 appears to have an exponential decrease for costs per individual trained, while the AN/BYG–1 exhibits a similar shape for costs per unit in service. In all cases, the costs per individual trained seemed to decrease as the number of trainees increased.

In addition to analyzing the costs of specific programs and systems, an analysis of software development labor costs was conducted. Labor is the primary cost driver in software development. Any trends in the costs associated with labor may also have significant influence on the cost of software development and maintenance. The national mean hourly wages for three software related occupations were analyzed from 1999 to 2007. The wages associated with each occupation were not consistent and fluctuated in both the upward and downward direction. Despite the year-to-year variations, they have increased from their 1999 levels.

Ultimately, the analysis and results do not support or refute whether reuse occurs and whether cost avoidance has been achieved. The data gathered and analyzed simply illustrate trends in the costs of systems and programs associated with the reuse of ASW software. Further, the upward trend in the cost of labor associated with software development illustrates that any downward trends in costs are influenced by other factors. Although the extent and specific levels of reuse across the programs were not determined,

the downward trends exhibited may represent cost savings that can be attributed to reuse. However, the data analyzed are not indicative of direct savings from software reuse.

## B. CONCLUSIONS

Software reuse may be successful in reducing the costs associated with the programs selected for analysis. Even as the cost of labor, the primary cost driver in software development and support, has increased, the costs associated with developing and maintaining anti-submarine warfare software have shown indications of reduction. Although specific empirical data concerning the savings from ASW software reuse was not available, some of the costs that were analyzed exhibited downward trends. However, trends in either direction could not be directly correlated to software reuse.

Any direct conclusions associated with software reuse are also complicated by the number of factors that can affect funding within the Department of Defense. During the course of the research, examples such as reprogramming of funds and congressional additions were found to have significant effects on annual funding. Therefore drawing meaningful relationships between the funding for two programs is difficult due to the influence of factors other than product development.

Also, it was not possible to consider in this research the specific capabilities that were developed. The only caveat for analysis was a relationship to the anti-submarine warfare mission. Although specific capabilities may have been a factor in the funding, they were not a factor in the analysis. This research indicates that the Navy does seem to be achieving some cost reductions related to ASW software maintenance and RDT&E. Although the reductions could be indicative of process improvements, they may also be caused by changes in priorities or an inability to provide adequate desired levels of funding.

Clements et al., (2006) state that “PEO IWS has been very successful in achieving software reuse at a level that may be unparalleled in the Navy” (p. 39). As of 2005, reuse of ASW software assets has been credited with saving the development of 4.5 million lines of code at an approximate cost avoidance of $500 million (Clements et al., 2006, p.

29). However, there does not seem to be a mechanism currently in place to track the actual cost savings obtained from reuse. Contractor experience is a factor in the acquisition process. This implies that reuse is a consideration in awarding contracts, but does not provide an indication of actual reuse levels.

The practice of reusing software is discussed throughout the literature, but there is a lack of consensus on the most effective methods available to take advantage of it. Large scale acquisition of complex systems makes a definitive answer on how to best achieve cost savings through software reuse even more difficult. Contributing to this difficulty is the continued evolution of the practice. For example, no specific programming language is used universally.

Although the focus of this research was on the benefits of reuse, there can also be instances when reuse becomes more expensive. Software programming languages are continually evolving and overreliance on past efforts can lead to exclusion of the benefits of new developments and advances in technology. As the process of software development evolves and matures, it is important that the acquisition community does not fixate on a single solution. Collaboration between organizations such as Communities of Interest may help alleviate some issues associated with software reuse, but their success still depends on active participation by all members.

Finally, concerns continue to exist over intellectual property (Clements et al., 2006, p. 31). The Navy and the Department of Defense should align their goals for software acquisition with a strategy that allows contractors to continually benefit from intellectual work. Policy and definitions must address both the concerns of the Navy and its contractors. Concerns over intellectual property should be addressed to ensure that the Navy continues to attract a variety of businesses and solutions for new combat systems.

## C. RECOMMENDATIONS

One of the issues mentioned in GAO (1993) regarding software reuse is the establishment of metrics. The report defines software metrics as “quantifiable measures that are used to assess the products and processes of software development” (p. 10). Future research should focus on defining metrics that illustrate cost avoidance or cost savings for the Navy.

There are numerous initiatives such as software repositories within the Department of Defense and Navy that focus on facilitating the reuse of software assets. Most of these require additional funding to maintain operations. There should be some mechanism in place to evaluate the performance of programs using a cost-benefit analysis. Current metrics seem to be focused on whether software assets are made available for reuse and not whether they result in cost savings while achieving mission requirements.

This research identified several programs that have illustrated cost reductions. Further research should investigate whether the reductions in costs have resulted in a corresponding reduction in capabilities. If costs are being reduced along with system capability, then it would be inaccurate to classify savings as cost avoidance. Future research should focus on a more limited number of programs. This could facilitate more meaningful analysis of the specific origin of the reduced costs.

Analysis of training costs on a per unit basis is not a valuable source of information during periods of transition to a common system. Any future analysis should focus on total fixed costs of operating training centers. If savings are being achieved, then the fixed costs should decrease. Analyzing costs on a per unit basis during periods of transition may only be a reflection of reallocating existing costs. Although the overall costs may not be changing, they may appear to rise due to the allocation of existing training courses to the new systems.

Future research should focus on more specific empirical data related to software reuse. The costs that were available for this research were too general to provide specific data related to cost avoidance from software reuse. If there is a necessity or desire to analyze the actual cost avoidance due to software reuse, additional measures should be taken. There is currently no central location for specific data related to ASW software reuse. Some efforts, such as Software Resource Data Reports for ACAT I and ACAT IA programs, incorporate information on software reuse and may provide valuable data for future analysis, but the current scope of these data is limited.

THIS PAGE INTENTIONALLY LEFT BLANK

# LIST OF REFERENCES


ASSET. (2006, August 25). Submarine Sonar Cost Analysis Report (Rev. 10) [PowerPoint slides].

American Society of Military Comptrollers. (2008, October). Enhanced Defense Financial Management Training Course. Arlington, VA.

Benedict, J. (2005). The Unraveling and Revitalization of U.S. Navy Antisubmarine Warfare. *Naval War College Review, 58*(2), 92-120.

Bergey, J., Fisher, M., Gallagher, B., Jones, L., and Northrop, L. (2000, February). Basic Concepts of Product Line Practice for the DoD. Pittsburgh, PA: Carnegie Mellon Software Engineering Institute.

Boehm, B., Abts, C., Brown, A., Chulani, S., Clark, B., Horowitz, E., et al., (2000). *Software Cost Estimation with COCOMO II*. Upper Saddle River, NJ: Prentice Hall PTR.

Boudreau, M. (2006, October 30). *Acoustic Rapid COTS Insertion: A Case Study in Spiral Development*. Monterey, CA: Naval Postgraduate School.

Bureau of Labor Statistics. *Occupational Employment Statistics* [Data File]. Available from http://www.bls.gov/OES/.

Clements, P.C., Cohen, S.G., and Bergey, J.K. (2006, December). *U.S. Navy ASW Sonar System Software and the Advanced Processing Build*. Pittsburgh, PA: Carnegie Mellon Software Engineering Institute.

Cohen, S. (2001, April). *Case Study: Building and Communicating a Business Case for a DoD Product Line.* Retrieved May 12, 2009 from http://www.sei.cmu.edu/productlines/.

Cohen, S. (2003, July). *Predicting When Product Line Investment Pays.* Retrieved May 12, 2009, from http://www.sei.cmu.edu/productlines/.

Cohen, S., Dunn, E., and Soule A. (2002, September). *Successful Product Line Development and Sustainment: A DoD Case Study.* Retrieved May 12, 2009, from http://www.sei.cmu.edu/productlines/.

Congressional Budget Office. (2009, March). A Preliminary Analysis of the President's Budget and an Update of CBO's Budget and Economic Outlook. Retrieved May 4, 2009, from. http://www.cbo.gov/ftpdocs/100xx/doc10014/toc.htm.

Cummings, C., Gallo, M., Johnson, P., Marsh-Jones, B., and von Kuegelgen, J. (1998, February). *Software Development Estimating Handbook, Phase One.* Washington, D.C.: Naval Center for Cost Analysis.

Department of Defense. (n.d). *The Open Systems Joint Task Force.* Retrieved November 10, 2008, from http://www.acq.osd.mil/osjtf/overview.html.

Department of Defense. (2009, May 1). *Financial Management Regulation (FMR) 7000.14-R.* Washington, D.C.

Department of Defense Chief Information Officer. (2007, May). *Communities of Interest in the Net-Centric DoD: Frequently Asked Questions (FAQ)* (Version 1.1). Washington, D.C.

Department of the Navy Financial Management & Comptroller. *Department of the Navy Budget Materials* [Data File]. Available from http://www.finance.hq.navy.mil/FMC/.

Department of the Navy Financial Management & Comptroller. (2003, February). *Department of the Navy Fiscal Year (FY) 2004/2005 Biennial Budget Estimates: Research, Development, Test & Evaluation, Navy.* Retrieved May 26, 2009, from http://www.finance.hq.navy.mil/FMC/.

Department of the Navy Financial Management & Comptroller. (2007, February). *Department of the Navy Fiscal Year (FY) 2008/2009 Budget Estimates: Research, Development, Test & Evaluation, Navy.* Retrieved May 26, 2009, from http://www.finance.hq.navy.mil/FMC/.

Department of the Navy Financial Management & Comptroller. (2008, February). *Department of the Navy Fiscal Year (FY) 2009 Budget Estimates: Research, Development, Test & Evaluation, Navy.* Retrieved May 26, 2009, from http://www.finance.hq.navy.mil/FMC/.

Government Accountability Office. (2008, September 25). *Defense Acquisitions: Fundamental Changes are Needed to Improve Weapon Program Outcomes.* Washington, D.C.

Government Accountability Office. (2004, March). *Defense Acquisitions: Stronger Management Practices are Needed to Improve DOD's software Intensive Weapon Acquisitions.* Washington, D.C.

Government Accountability Office. (1993, January 28). *Software Reuse: Major Issues Need to Be Resolved Before Benefits Can Be Achieved.* Washington, D.C.

Government Accountability Office. (1989, March). *Submarine Combat System: Technical Challenges Confronting Navy's AN/BSY-2 Development*. Washington, D.C.

Haefliger, S., von Krogh, G., and Spaeth, S. (2008, January). Code Reuse in Open Source Software. *Management Science*, *54*(1), 180-193.

Held, T., Newsome, B., and Lewis, M. (2008). *Commonality in Military Equipment: A Framework to Improve Acquisition Decisions.* Arlington, VA: RAND Corporation.

Johnson, W. (2004). The A-RCI Process—Leadership and Management Principles. *Naval Engineers Journal*, *116*, 99-105.

Karlsson, E-A. (1995). *Software Reuse: A Holistic Approach*. New York: John Wiley & Sons.

Lockheed Martin. (2003). USS Virginia Class C3I System. Retrieved December 17, 2008, from http://www.lockheedmartin.com/data/assets/1082.pdf

Mili, H., Mili, A., Yacoub, S., and Addy, E. (2002) *Reuse-Based Software Engineering Techniques, Organization, and Controls*. New York: John Wiley & Sons.

Naval Center for Cost Analysis. *Navy Visibility and Maintenance of Operating and Support Costs (VAMOSC)* [Data file]. Available from http://www.navyvamosc.com/.

Naval Center for Cost Analysis. (2009, February 27). *Naval Visibility and Maintenance of Operating and Support Costs (VAMOSC) 8.0: Shipboard Systems User Manual.* Retrieved May 26, 2009, from http://www.navyvamosc.com/.

Naval Center for Cost Analysis. (2009a, April 30). *FY 94-08 Ships, Shipboard Systems, and Military Sealift Command Anomalies*. Retrieved May 26, 2009, from http://www.navyvamosc.com/.

Naval Meteorological and Oceanographic Command. (2009, February). *Navy Standard Oceanographic and Atmospheric Master Library (OAML)* [Powerpoint Slides].

Naval Sea Systems Command PMS 4252. (1999, January 8). Acoustic Program Plan (Revision 8). Arlington, VA.

Nazareth, D. and Rothenberger, M. (2004, October). Assessing the Cost-Effectiveness of Software Reuse: A Model for Planned Reuse. *The Journal of Systems and Software*, *73*(2), 245-255.

Nelson, E. (2008, September 30). *Open Architecture Technical Principles and Guidelines 1.5.8.* Retrieved November 9, 2008, from https://acc.dau.mil/CommunityBrowser.aspx?id=170302&lang=en-US

Open Systems Joint Task Force. (2004, September). *Program Manager's Guide: A Modular Open Systems Approach (MOSA) to Acquisition, Version 2.0*. Washington, D.C.

Prieto-Diaz, R. and Freeman, P. (1987, January/February) Classifying Software for Reusability. *IEEE Software, 4*(1), 6-16.

Program Executive Office—Integrated Warfare Systems. (2007, June 21). Charter for the Anti-Submarine Warfare (ASW) Community of Interest (COI). Washington, D.C.

Program Executive Office—Integrated Warfare Systems 5A. (2003, July 11).APB Process Operating Instruction. Washington, D.C.

Program Executive Office—Integrated Warfare Systems 5. (2009, January 29). Management of the Advanced Development Process for Submarine Combat Systems. Washington, D.C.

Program Executive Office—Integrated Warfare Systems 7. (2007, October 25). *Naval Open Architecture Contract Guidebook, Version 1.1*. Washington, D.C.

Poulin, J.S., Caruso, J., and Hancock, D. (1993). The Business Case for Software Reuse. *IBM Systems Journal, 32*(4), 567-594.

Rothenberger, M., Dooley, K., Kulkarni, U., and Nada, N. (2003, September). Strategies for Software Reuse: A Principal Component Analysis of Reuse Practices. *IEEE Transactions on Software Engineering, 29*(9), 825-837.

Schade, O. (2009). *99 Bottles of Beer*. Retrieved May 19, 2009 from http://99-bottles-of-beer.net/.

Software Engineering Institute. (2009). Software Product Lines. Retrieved May 5, 2009, from http://www.sei.cmu.edu/productlines/.

Software Resource Data Report. (2005, October 25). In *Defense Acquisition Guidebook online*. Retrieved May 22, 2009 from https://akss.dau.mil/dag/DoD5000.asp?view=document.

# INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Dr. Joseph G. San Miguel
   Naval Postgraduate School
   Monterey, California

4. Michael W. Boudreau
   Naval Postgraduate School
   Monterey, California

5. RADM Steven Johnson
   Director, Strategic Systems Programs
   Arlington, Virginia

6. CAPT Charles Davis
   Program Executive Office Integrated Warfare Systems 5
   Washington Navy Yard, Washington DC